MASTER COPY KEEP FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 24, 1998	Final Progress Report Aug.1,97-Aug.31,98

4. TITLE AND SUBTITLE
Combat Simulation Trajectory Management

5. FUNDING NUMBERS
DAAG55-97-1-0360

6. AUTHOR(S)
John B. Gilmer Jr. and Frederick J. Sullivan

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
Wilkes University
P.O. Box 111
Wilkes-Barre, PA 18766

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 37434.1-MA

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The goal was to assess the scalability of the multitrajectory simulation technique, which earlier had been tested for scenarios of 40 units or less. More effort was needed to overhaul, verify, and port the "eaglet" simulation than expected, preventing exploration of the more advanced techniques for trajectory management that were planned. Automated methods for generating scenarios of various sizes were developed. Scenario outcomes were mapped to a Measure of Effectiveness space of Blue Losses vs. Loss Exchange Ratio. For each scenario, a 5 million stochastic run set was used to establish a reference. For multitrajectory and stochastic techniques, a measure of distance between the MOE plots and the reference was found. The multitrajectory technique showed a superiority ranging from marginal in the 4 division (60 unit) scenario to a factor of 4 better in the larger 8 division (120 unit) case. There were troubling indications of unexplained problems. In a small two division case, the multitrajectory algorithm unexpectedly performed more poorly. More sophisticated algorithms, such as the "depth first" multitrajectory mode, proved beyond the scope of what was possible.

14. SUBJECT TERMS	15. NUMBER IF PAGES
Simulation, Combat, Multiple Trajectory, Stochastic, Nonmonotonicity	39
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Combat Simulation Trajectory Management

Final Report

John B. Gilmer Jr.
Frederick J. Sullivan

Wilkes University
P.O. Box 111
Wilkes Barre, PA 18766

November 24, 1998

Table of Contents

List of Appendices

List of Illustrations:

List of Tables:

Abstract:

The goal was to assess the scalability of the multitrajectory simulation technique, which earlier had been tested for scenarios of 40 units or less. A large amount of effort was needed to overhaul, verify, and port the "eaglet" simulation and simulation support software that were used for the scaling experiments. This required considerably more effort than expected, preventing exploration of some of the more advanced techniques for trajectory management that were planned. Automated methods for generating scenarios of various sizes were developed. Scenario outcomes were mapped to a Measure of Effectiveness space of Blue Losses vs. Loss Exchange Ratio. For each of several scenarios of increasing size, a huge number (5 million) of stochastic runs was used to establish a reference MOE plot, against which results of other runs could be measured. For multitrajectory and stochastic techniques, a measure of distance between the MOE plots and the reference was able to establish correspondence between the number of stochastic replications and multitrajectory trajectories giving equivalent performance. The multitrajectory technique showed a superiority ranging from marginal in the 4 division (60 unit) scenario to a factor of 4 better in the larger 8 division (120 unit) case. (Larger cases of 16 and 32 divisions were run, but resources did not allow time to develop a reference plot for comparison.) There were troubling indications of unexplained problems. In a small two division case, the multitrajectory algorithm seemed to produce a slightly different set of outcomes than in the stochastic case, and only did better by the distance metric with heavy smoothing was used to eliminate fine grained randomness. More sophisticated algorithms, such as the "depth first" multitrajectory mode, proved beyond the scope of what was possible. These would have helped resolve the unexplained features of our results, and would probably have given superior results. In conclusion, the multitrajectory technique seems to perform better relative to purely stochastic simulation as the scenarios scale to larger sizes. However, there are enough unexplained issues to require considerable caution in interpreting these results.

Statement of the Problem Studied:

The Multitrajectory Simulation technique is believed to address some of the practical problems for analysis caused by non monotonic behavior of large scale combat simulations of the kind used by the U. S. Army Concepts Analysis Agency and others. Earlier research showed that the technique has promise for conveying more information at less cost than traditional stochastic simulations. But those results were based on scenarios much smaller than those used for typical combat analysis. The goal of this project was to better understand the potential of multitrajectory simulation at a scale more directly applicable to that needed for military analyses.

This research, and that which preceded it, has been conducted using a simplified, unclassified surrogate for the simulations of interest. The Eagle simulation currently used by CAA has been used as an example of the capabilities essential to combat simulations. Our surrogate for Eagle, "eaglet," includes similar characteristics, though in simplified form. It features autonomous units which move along routes, engage in combat, and make decisions concerning operational posture and changes to objectives. The "eaglet" simulation implements Multitrajectory Simulation: it incorporates mechanisms for spawning separate simulation trajectories at critical events associated with movement selection events, acquisition events, decision breakpoints and command/control (C2) behavior. A more complete description of "eaglet" was given in the final report for the previous project, and other papers published based on that work. [references]

In the original research effort, two versions of "eaglet" were developed. The first used an "ad-hoc" implementation of multitrajectory simulation, which was embedded in the functional modules (those implementing modeled behavior of "move", "decide", etc.). This version came to be called "Gilmer's method" for multitrajectory implementation. It was intended to be a prototype that would illustrate the functionality that was needed and provide some early results.

An improved implementation reorganized the simulation code to use "Base Classes", to make the multitrajectory features nearly transparent to programmers writing the model function code. This version became known as "Sullivan's Method." However, there were problems in this implementation, and it proved to work properly only for a limited subset of cases we hoped to study. As a result, all of the analytic results for the previous study were produced using the original prototype.

The shortcomings in the operation of the "Sullivan's Method" (or "base classes") version needed to be rectified. This was a high priority for the project, and one for which the necessary effort was not appreciated at the time the proposal was made. Indeed, the software rework necessary to do this consumed the majority of effort available for the project. This will be discussed in more detail later.

Software techniques for maintaining diverging states were to be further developed as necessary to support larger scenarios. An assessment was to be made of whether trajectory merger (demonstrated in earlier research) was a practical prospect for cases having large numbers of states. Better metrics were seen as needed for state distance estimation. Comparison to statistical methods for a large scenario was to be based on the coverage of the range of outcomes. An assessment was then be made on the practicality and benefits of implementing such trajectory management measures as modifications to simulations in the Army's inventory.

As will be described later, we were not able to do all of this to the extent we had hoped. Scalability experiments were conducted, but the use of certain more advanced multitrajectory techniques proved to be beyond the time and resources available. In particular, we were not able to exercise "state merger" options.

The tasks identified for this project were:

1.1 Port of "eaglet" to SGI platform (and also: Get it running correctly!)
1.2 Develop and implement improved MT techniques
1.3 Develop large scenarios
2.1 Make runs and characterize for "simple" MT
2.2 Make runs, characterize, for "advanced" MT
2.3 Develop measures, assess vs. stochastic

The project budget was $47681, provided by grant DAAG55-97-1-0360 from the U. S. Army Research Office.

Summary of the Most Important Results:

The data collected showed that as scenario size increases, the Multitrajectory algorithm (actually, one that begins in multitrajectory mode and transitions to stochastic mode as the resource limit is reached) performs better than a purely stochastic mechanism. For a two division sized scenario, the multitrajectory technique did no better, ands was actually somewhat worse. For the four division case, the metric used to compare closeness to the definitive outcome was four times better for multitrajectory than stochastic at 10,000 replications. Put differently, about 70,000 stochastic replications would be needed to compare with 10,000 multitrajectory replications. For the 8 division scenario, at 10,000 replications the multitrajectory technique was better by a factor of 2 in the metric distance. The stochastic data was decreasing so slowly that even 1,000,000 stochastic replications did not match the multitrajectory results.

However, there were enough unexplained features in the data that at this time these results cannot be entirely trusted. In particular, the two division scenario was actually expected to do much better, and some irregularities indicate there may have been some execution error that

widened the discrepancy between multitrajectory and stochastic results. We had expected the multitrajectory technique to do worse with scenarios of increasing size, and are unable to explain why the multitrajectory approach did better in the four division scenario than in the two division scenario.

Status:

The accomplishments of this project include the following:

1. "eaglet" and Multitrajectory (Sullivan's Method) implementation were overhauled.
2. "eaglet" was ported to Silicon Graphics Inc. (SGI) and Linux (Intel PC) platforms.
3. Verification testing vs. the Mac version (Gilmer's Method) of "eaglet" was completed.
4. Incoherent and Hierarchical scenario generators were developed and implemented.
5. Data collection and analysis methodology was developed, and software completed.
6. Analytic runs of up to 8 "divisions" size were made, data collected, and analyzed.
7. Scaling experiments with different event management techniques were conducted.
8. A better understanding of issues, potential techniques, and alternatives was developed.

In our experiments with trajectory management, the techniques we actually explored included the following:

1. Restriction of event set resolved as MT
2. Methods that go stochastic at a fixed state limit
3. Methods using a "soft" state threshold, beyond which events resolution is stochastic.

The methods which we hoped to use, but were not able to get to, included:

1. The use of Measure of Effectiveness (MOE) sensitive event resolution
2. Pruning techniques, based on state probability and MOE criteria
3. Better distance estimation metrics

The MOE sensitive technique was investigated by a graduate student on an unfunded basis after the end of the previous project and before the beginning of this one. He was able to produce some interesting results, but had only been able to get the small (4 unit) scenario working at that time with the Sullivan's Method version of "eaglet". We had hoped to see the scaling performance of this particular approach, but that remained beyond our grasp. The same student had also explored state merger in the same small scenario context, which we had hoped to extend to larger scenarios. However, basic changes needed to handle the larger numbers of trajectories in the larger scenarios proved beyond what we could accomplish. His results have been published. [reference]

The primary reason for our making less than the expected progress was an underestimate of the condition of the "Sullivan's Method" version of "eaglet". Development of this version within the previous project was actually an accomplishment beyond the scope of what we had originally proposed. We knew that there were some problems remaining, yet it had proved usable enough for the research in MOE sensitive trajectory management. As it turned out, many of the problems had been patched over, and a major reorganization of the base classes proved necessary. As a consequence, we spent 75% of the project duration "fixing" the Sullivan's Method version of eaglet. Other useful work was performed concurrently, but this was still a big resource drain.

Scenario Generation Requirements:

It was clear from the beginning of this project that manual generation of large scenarios would be impractical. Some form of automated scenario generation would be needed. Yet, the scenarios would need to be realistic to the extent that the behavior of events would be similar to what one might experience in the use of an analytic grade simulation.

A few years ago when the scaling issue was first discussed, the possibility of simply making multiple copies of a given small scenario (with spatial displacements) was suggested as a way to generate larger test scenarios. The problem with this is that units in the various copies will be perfectly synchronized, and the behavior of the simulation will be consequently unnaturally subject to peaks and lags of activity. This would have significant consequences for the multitrajectory behavior. A more realistic, or at least less coherent, approach was needed. The concept developed to fill this need, based on a hierarchy of templates for generating unit laydowns and tasks, is a simplification of "frames" (or template) based planning. This became the "hierarchical" planner, or scenario generator.

However, such a hierarchical planner would not be able to generate scenarios of any arbitrary size n (with n being number of units) with n being continuously scalable. Thus, there was a need for a simple method for large scenario generation (which could be easily understood and implemented by the students with a minimum of supervision). This method would not reflect a hierarchical unit structure, but would be useful for certain experiments where n scales continuously, and only unit events like movement and acquisition were being exercised. This became the "incoherent" planner. The incoherent planner was completed too late to be used in the analysis that was performed, so it is described in Appendix A rather than in the body of this report.

The Hierarchical Planner

In "eaglet", units have no functional hierarchy, even though there is a command structure in the tree of HQ units. The HQ units in fact only perform an "operate" operation, which has no effect on other units. The function of command is implicit in the tasks with which other units begin the scenario. This simplification was necessary and reasonable in the initial "eaglet." It was hoped that a more functional representation of decisionmaking would be added as part of this project. The improvements made were not as extensive as originally planned. Nevertheless, the hierarchical planner as a tool for generating scenarios takes advantage of the command structure to generate and tasks units that compose a higher echelon unit. The planner accepts a scenario having aggregated units of higher echelons. If one or more plan template exists for the unit for the mission assigned, that unit is decomposed into its constituent units in accordance with the template, and the subordinates tasked as the template directs. This is done hierarchically, so that one aggregated unit in the original scenario will cause multiple levels of subordinates to be generated.

For the analyses we performed, templates were developed for Blue attack operations at the brigade and division levels. Templates for Red defend operations were also developed for brigade and division levels. The original scenario file used has pairs of Blue and Red divisions, with the Blue divisions attacking the respective Red ones. This parallel frontal attack is against all logic for realistic practical scenarios, but does give us a feasible method for generating larger scenarios. In a sense, these will behave worse (with respect to problems generated for multitrajectory simulation) than a more realistic large scenario, in which combat is more episodic. Use of multiple templates for the same unit echelon and operation, and a Corps level template, may have given more realistic results, but exceeded the effort available.

The hierarchical planner uses, in template form, a representation of an operation plan for a unit that is disaggregated. This is logically an elaboration of the Task object in "eaglet". A Task turns into a Plan when disaggregation takes place. (The planner is built on the same structures as "eaglet", which may allow integration of planning into "eaglet" if needed at some time in the future.) While a task includes only the information necessary for one unit (task type, intent, operational activity, objective, and such) a plan must include organization of subordinates, assignment of them into roles, and phasing information. The latter define how one phase of the planner's superior's plan is implemented in perhaps multiple phases of the planning unit's plan. (We did not implement phase objects, since the "eaglet" C2 representation does not use them.)

"Subordinate" role objects are needed to define what each subordinate would be doing in the plan. Other kinds of roles, for enemy units and terrain, were avoided given our purposes. ("eaglet" has no terrain, a short battle duration, and is not expected to produce analytic quality results, so the fidelity advantages and sophistication of using enemy unit roles is not worthwhile. Roles have template units and tasks that are instantiated when the plan (or, more properly, the plan template) is used to create a disaggregated force. This process changes the aggregated unit's original task into a plan, with units having appropriate tasks. An illustration of the plan template structure is shown below:

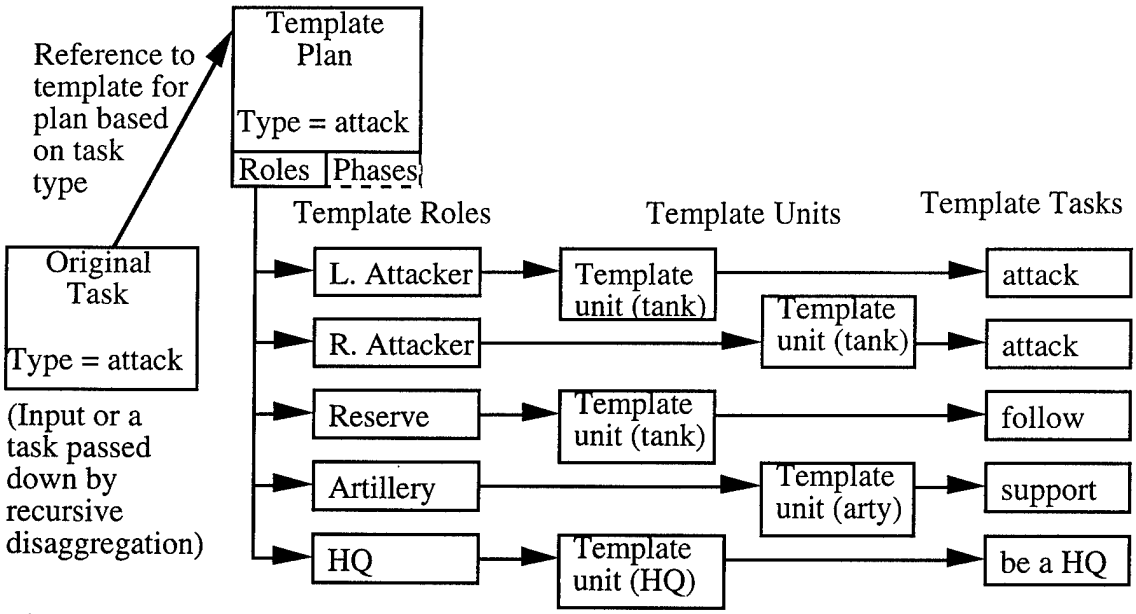


Figure 1 Templates Developed for Attack and Defend, for the Brigade and Division level.

Units that are generated are given routes, which are also included in template form. A simplified example set of routes for a reserve unit in a brigade attack is shown below. As defined, all of the routes used have at least two paths, though this is not shown in the figure. Routes are also represented in template form, similar to those in the Incoherent Planner.

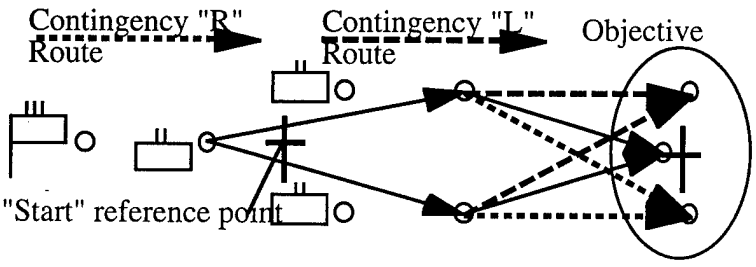


Figure 2 Attack Regiment Echelon Operation Template with Contingencies

The placement of subordinate unit initial laydowns, objectives and intermediate route waypoints are varied randomly governed by a parameter that defines the scope of variation (by default, 25%). Thus, different divisions and brigades in a scenario are not executing exactly the same plan; each will be somewhat different due to these differences in placement of points, giving variations in events.

Trajectory Management

The most basic problem in Multitrajectory Simulation is the threat of combinatorial explosion in the number of trajectories. This is especially important as a scaling issue, since a manageable number of trajectories for a small scenario will become unmanageable for larger scenarios if no change is made in the way events are handled.

In general, as the number of units n grows large, the probability of a given trajectory, p(state), approaches zero (we can get underflow with doubles precision). Also, the number of events approaches kn if a typical unit has k events, and (worst) the number of trajectories rapidly approaches infinity, or 2^{nk}, if all events are binary. The figure below illustrates this.

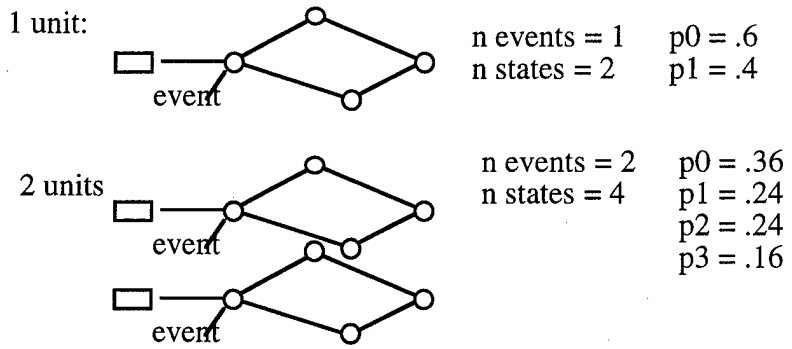


Figure 3 Scaling Effects Example for Route Events

The most basic trajectory management technique that was used was first explored in the earlier project was used extensively in this one as well, and has given us most of the data presented later. This is a combination of Multitrajectory and Stochastic approaches. The multitrajectory technique is used up until the memory budget (limit to the number of states) is expended, and further events are resolved stochastically. In the previous project, we saw that this seemed to give significantly better results than a purely stochastic simulation. However, that effort did not quantify the benefit, nor did it show how the benefit might scale for larger scenarios.

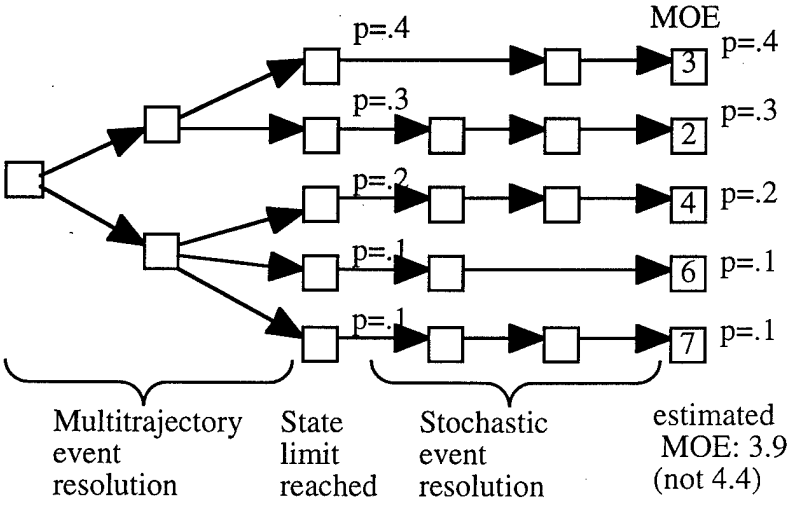


Figure 4 Multitrajectory - Stochastic Combination Method

Up until the state limit is reached, we have exact knowledge of the outcome space, including the relative probabilities of the trajectories. (By "exact", we mean given the simulation, scenario, and choice of events to examine by allowing nondeterministic choice. Of course, none of these assumptions of fidelity, representativeness of the scenario, and choice of events is fully justified, but is a compromise that the analyst willingly makes to reduce the overall problem to a manageable level that can be addressed with the tools at his disposal. Multitrajectory simulation needs to be understood in that context.)

When the limit is reached and stochastic resolution begins, we can continue to track the probabilities of the individual states, and can know their true probabilities. The sum of these probabilities rapidly decreases, and thus the coverage of the outcome space declines markedly. The actual coverage may be useful to know. But for purposes of estimating the nature of the outcome space, for example with respect to some MOE of interest, we will want to leave the trajectory probabilities unchanged, since the trajectory is now the sole surviving representative of its slice of the outcome space, as determined at the time the state limit is reached. Had we chosen to resolve events deterministically after the limit is reached, we would maximize the outcome space coverage, but the set of final outcomes is less representative than continuation with stochastic processing. (This was an important finding from the earlier project.) The degree to which this is true increases the earlier the state limit is reached. Important later events, if resolved only in deterministic fashion, cut the analyst off from an entire family of outcomes that often are quite important. Understanding this benefit quantitatively was an important goal of the project.

We believe that the benefit of this approach can be determined using theory for relatively simplified cases. The difference in scale and complexity for combat simulations, and the prescribed scope of this study, has not yet allowed development of this theoretic understanding. We hope to do that in the future.

The approach above causes all events prior to a given time to be resolved in multitrajectory fashion, and all later ones to be stochastic. If the early events were the most important ones, this would perhaps be ideal. However, that will not often be the case. In the largest scenarios, with a low state limit, the only events that may be resolved as multitrajectory are the initial movement events of the attacking units. A seemingly good hypothesis is that upper level C2 events, such as the commitment of a reserve unit, ought to the most important, and thus the highest priority for multitrajectory treatment. But since they come later in time, the state budget has typically already been reached. Indeed, as scenario size grows, one would expect that individual unit events would fade in significance as the larger numbers of these events pushes the aggregate behavior toward some kind of mean.

The discussion above suggests that, as scenario size grows, the selection of an event set ought to be successively restricted to preserve a priority for those that are most important. Given the breadth first execution mode of "eaglet", there is no really good way to make these decisions within the running simulation dynamically. (With the "depth first" or a hybrid execution mode to be described later, this may well be practical.) Figure x below illustrates how the number of events per division scales. Our goal would be to keep the total number of events approximately constant. Our hypothesis is: As n gets larger, we can restrict the event types for which we use Multitrajectory Simulation and still have an "acceptable" effect on the outcome set (with respect to a given MOE distribution), relative to stochastic simulation. We have not yet collected data necessary to test this hypothesis.

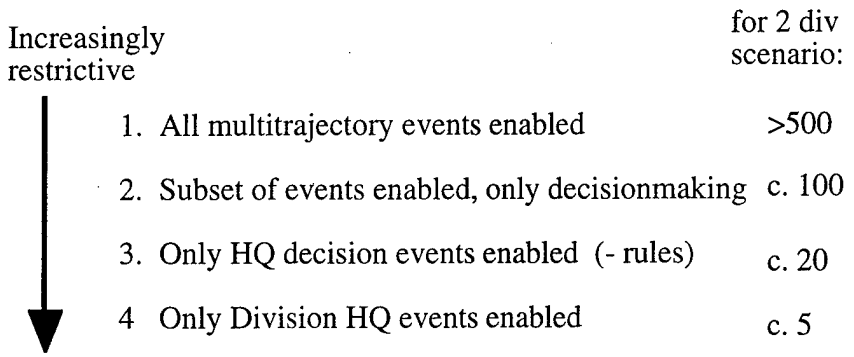


Figure 5 Selection of Smaller Event Subsets for Trajectory Management

One strategy for distributing the multitrajectory resolution of events over a greater range of times is the "Leftist Tree" run mode. This was originally developed to allow a given trajectory to be repeated by following the event resolutions previously recorded, but with graphics to show the evolution of the trajectory. At each event, a multitrajectory resolution is made, but only for this one trajectory. Each of these side trajectories does not branch further. Events in the side trajectories are resolves similarly to that of the main (or reference) trajectory, if that cam be determined, or in deterministic fashion otherwise. This method generates only one extra trajectory for each event, so the number of trajectories is small. The effect is similar to doing a sensitivity run for each variation of an event against the (reference) base case. However, this method does not generate trajectories that are sensitive to synergistic variations, and the number of trajectories will generally be too limited. On the other hand, every event gets some exercise in multitrajectory fashion. If all events are turned on, this should result in a fairly interesting sampling of the outcome space. A necessary precondition is the existence of an event record that the base case (or reference) trajectory follows. Variations which have not yet been explored include leftist tree runs using stochastic or multitrajectory references generated from other methods. We have not yet had a chance to try this method of sampling and compare it to other techniques. Figure 6 below illustrates the approach.

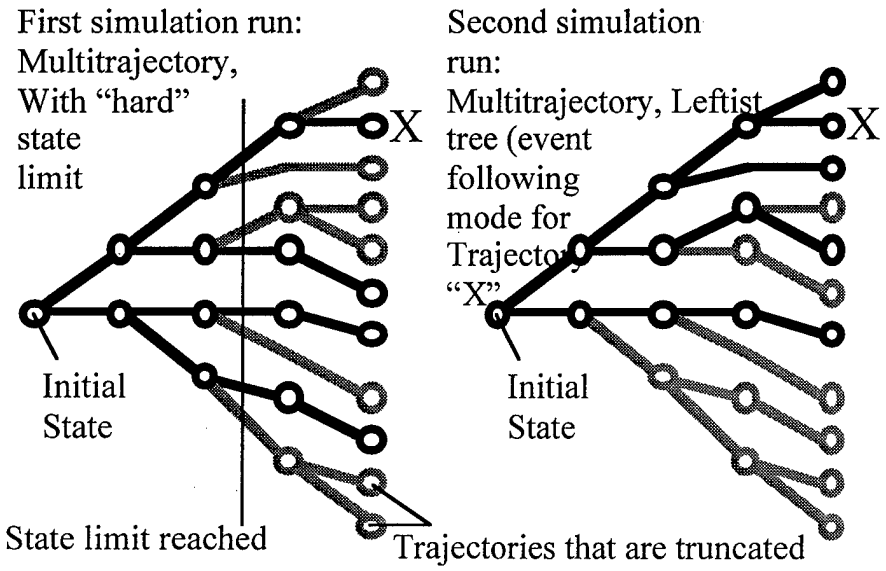


Figure 6 Leftist Tree Event Resolution Mode

Analysis

The analysis performed in this project was designed to compare the performance of the multitrajectory technique to that of traditional stochastic simulation. We have selected two Measures of Effectiveness to form a surface onto which the outcome space generated by a given set of runs can be projected. We then can compare various outcome space subsets produced by different methods by comparing their "pictures" formed by this projection onto the two dimensional MOE space. In particular, we wish to compare sets of n stochastic runs with the outcome of multitrajectory runs with a state limit of m, and find the values of n and m that give equivalent performance in terms of the fidelity with which the MOE plot is given. Figure x below illustrated the process:

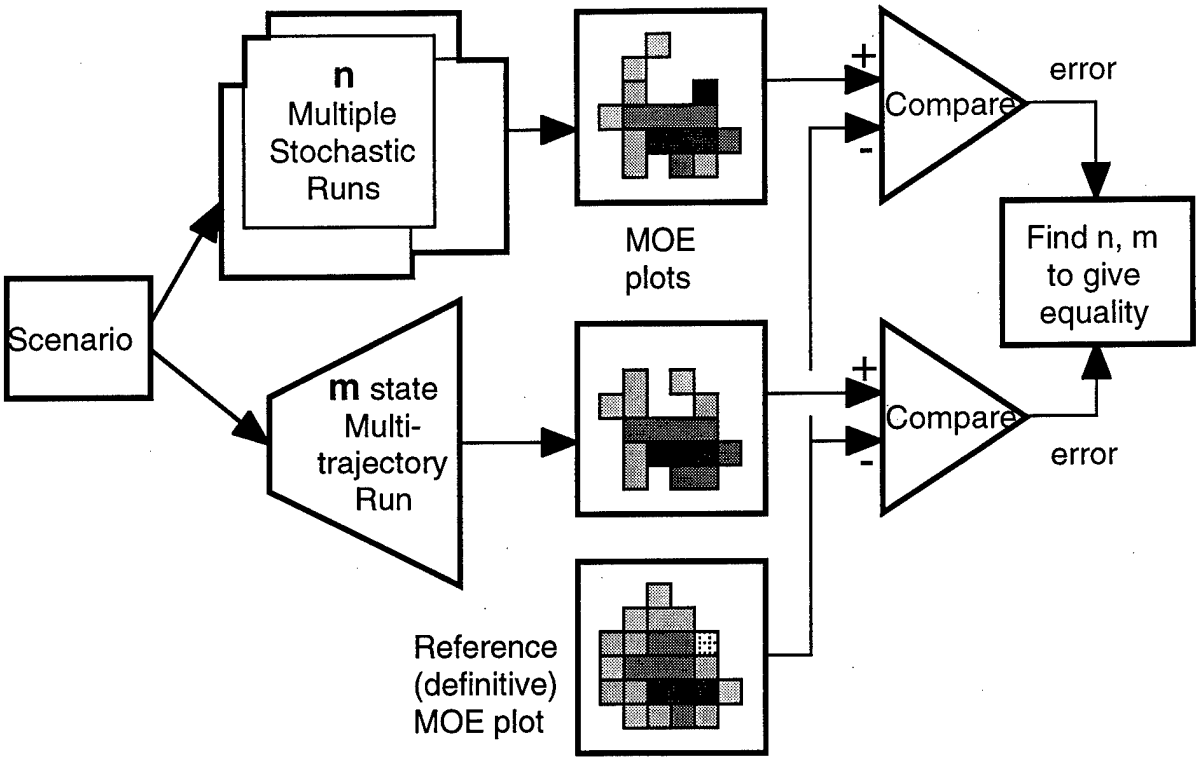


Figure 7 The Analysis Process Used to Compare Multitrajectory and Stochastic Runs.

The analytic approach described above requires as a definitive reference an "outcome set" against which comparisons can be made. For small scenarios, exhaustive breadth first execution in multitrajectory mode was possible for the simpler some cases, giving a complete outcome set. This was the standard used for the previous project, for the small 4 unit scenario, when the total number of trajectories was of manageable size. However, with the current "breadth first" implementations of "eaglet", memory bounds make large multitrajectory runs impractical. Virtual memory is not much of a help, as disk thrashing slows execution to a painfully impractical crawl. In order to get this definitive set, we resorted to huge numbers of stochastic runs. Generally, the number used was 5 million. As the number of runs increased, we checked to ensure that the differences in the MOE plot led to smaller and smaller changes, as described later.

Figures 8, 9, and 10 show the MOE plots for 1,000 to 5,000,000 replications for the two division scenario. The probabilities are scaled logarithmically. The white cells had no replications with that MOE value.

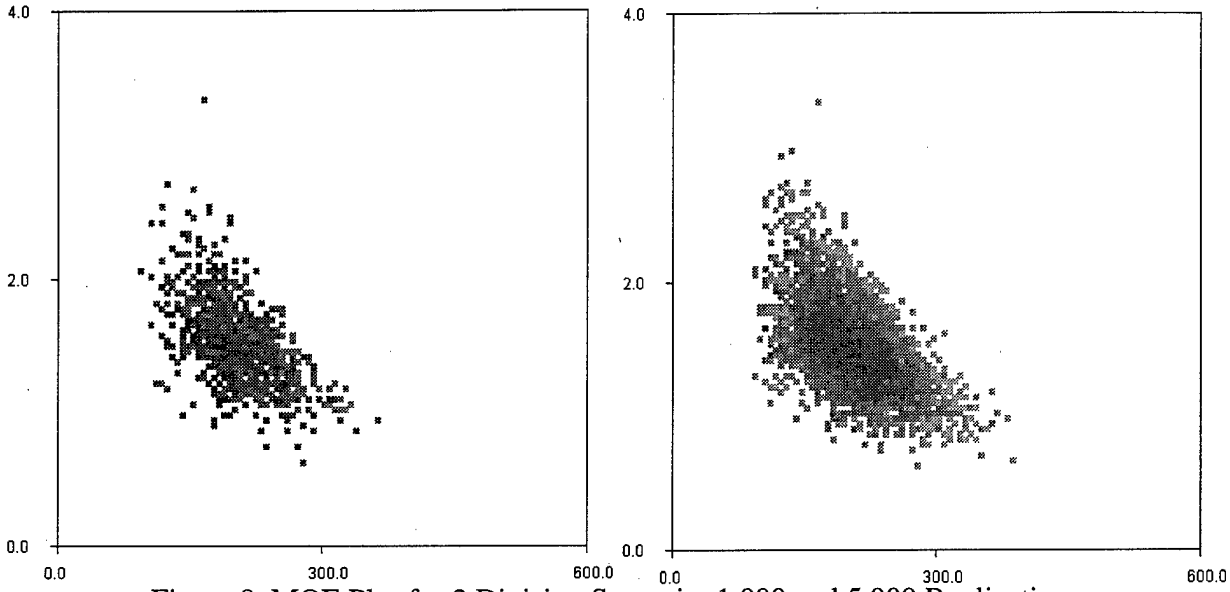


Figure 8 MOE Plot for 2 Division Scenario, 1,000 and 5,000 Replications

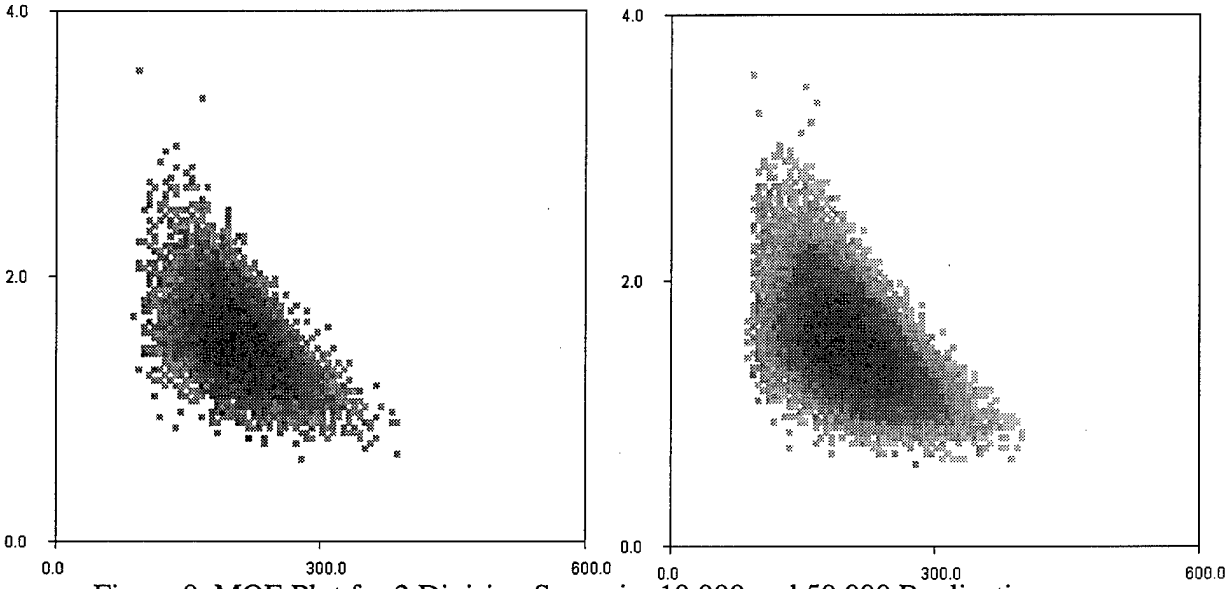


Figure 9 MOE Plot for 2 Division Scenario, 10,000 and 50,000 Replications

The technique for making MOE plot comparisons needs elaboration. Each MOE plot was a 100 x 100 matrix of cells ranging from 0 to 4.0 for Loss Exchange Ratio (Y), and from 0 to 600 for Blue losses (X). In each cell shading indicates the sum of the probabilities of all trajectories terminating with MOE values that fell in that cell. Before making a distance comparison between run sets, each plot was normalized so that the sum of all probabilities was 1.0. Then, to account for the possibilities of small lateral displacements in the MOE space and to smooth out the randomness in regions having few replications, the plots were smoothed by convolution with a 3x3 center weighted filter:

[[.0625, .125, .0625],[.125,.25..125],[.0625, .125, .0625]]

Then the sum of the differences between respective cells was taken. This measure of distance from the definitive set was, in effect, out figure of merit for a given set of simulation runs.

Only relative distances are really meaningful. Figures 11 and 12 illustrate the effects of smoothing MOE plots for the 1,000 and 10,000 replication cases. Without smoothing, the random nature of the data together with what amounts to excess resolution for the MOE dimensions give metrics that are not very meaningful. We also used an 11 x 11 smooth with each dimension scaled using a cosine distribution for weighting factors, when the 3 x 3 smooth proved inadequate for small numbers of replications. The convergence with increasing numbers of stochastic runs is illustrated for the two division scenario in Table 1 below. For this set of replications, all event types that were subject to variability in the simulation (movement, acquisition gain and loss, and decisionmaking) were made stochastic. On each line, the distance is given for the replication set compared with A set of 5,000,000 replications. The smoothed distances are achieved with the 3x3 filter given above. (Note: the very lightest shade of gray was lost in image translation, so a pair of very light splotches at the top of the 11 x 11 figure for 10,000 replications cannot be seen.)

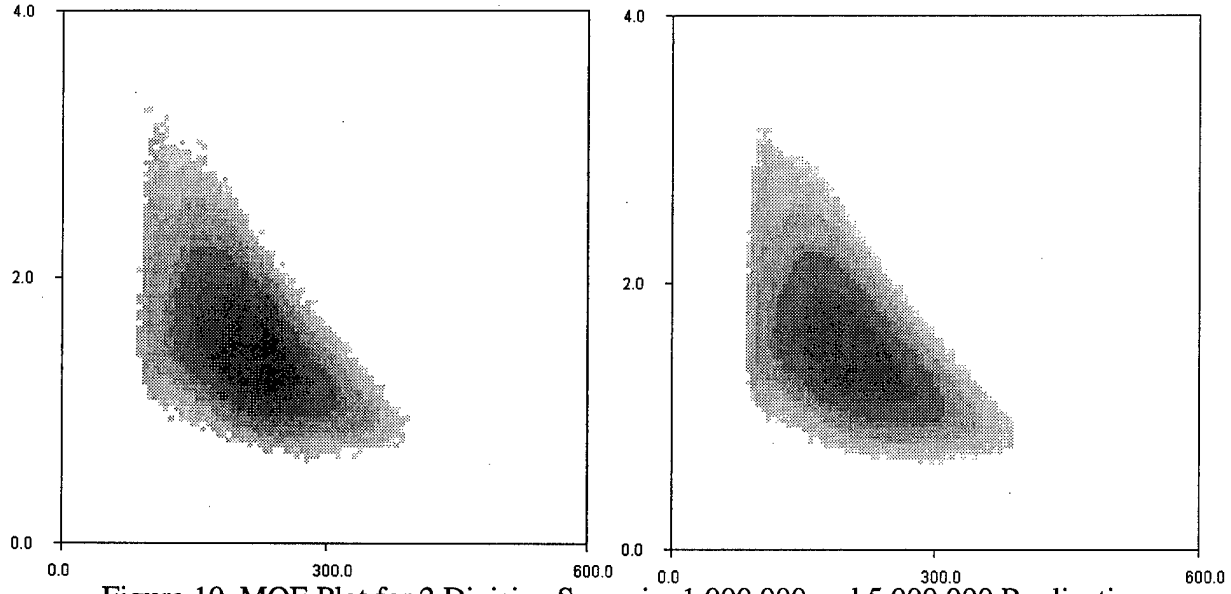
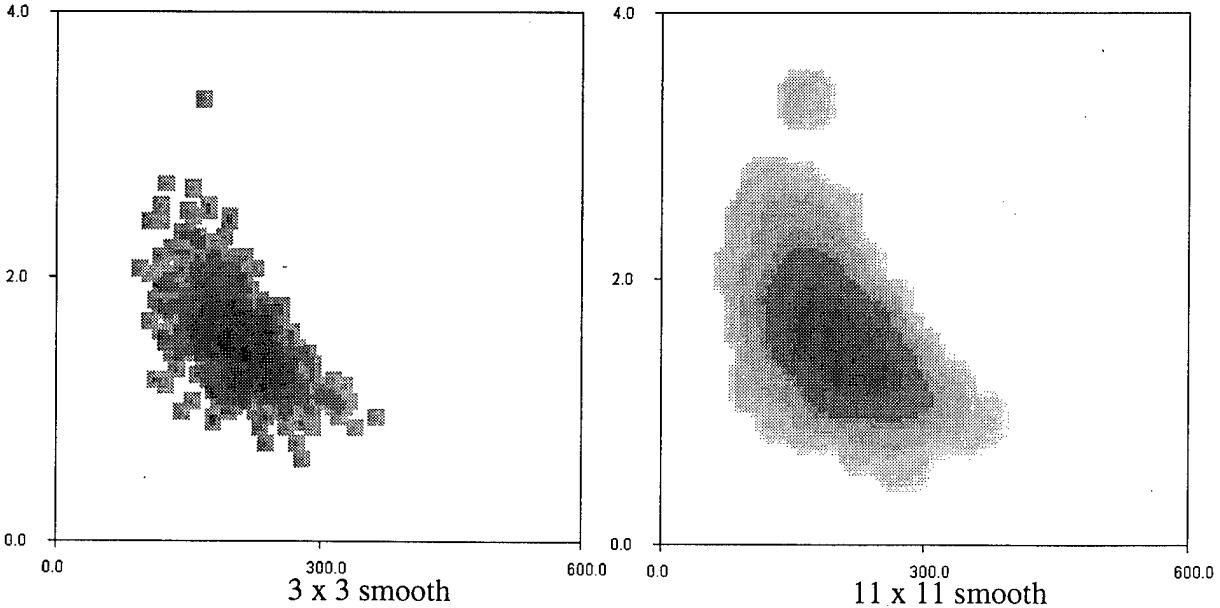


Figure 10 MOE Plot for 2 Division Scenario, 1,000,000 and 5,000,000 Replications



3 x 3 smooth 11 x 11 smooth

Figure 11 Effects of Smoothing on 1,000 Replication MOE Plots (for 2 Division Stochastic Case)

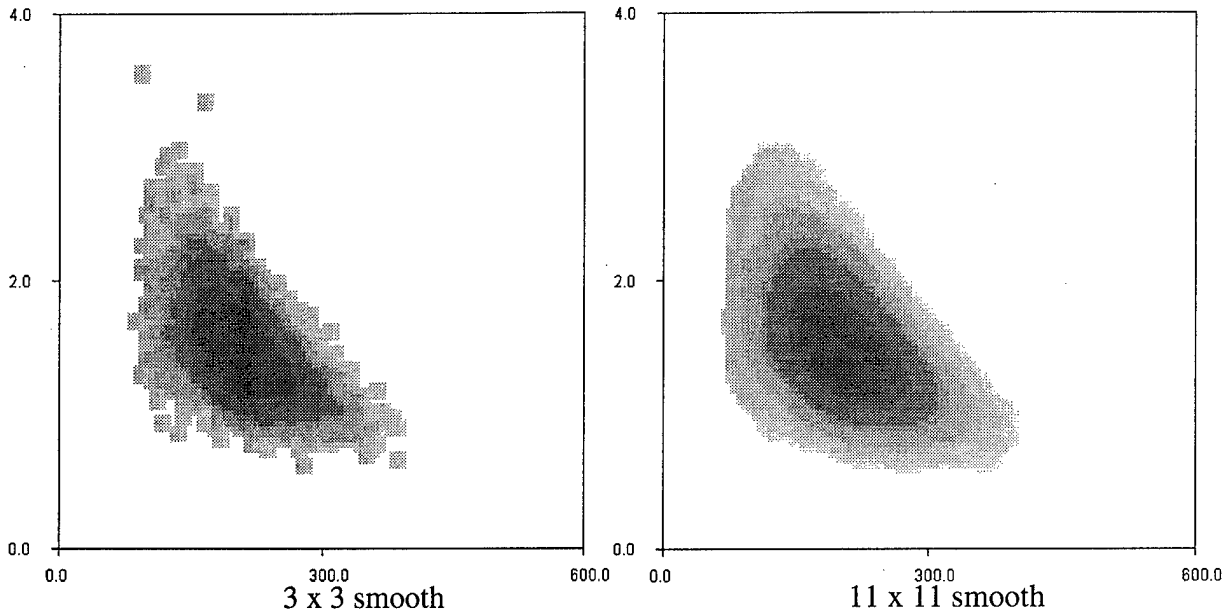


Figure 12 Effects of Smoothing on 10,000 Replication MOE Plots
(for 2 Division Stochastic Case)

Table 1: Two Division Stochastic Run Convergence data

	Unsmoothed Distances		Smoothed Distances	
Outcomes	Max	Average	Max	Average
1000	5.66E-03	7.28E-05	2.24E-03	2.99E-05
2000	3.30E-03	5.52E-05	1.31E-03	2.35E-05
3000	2.51E-03	4.47E-05	1.06E-03	1.96E-05
4000	2.94E-03	4.01E-05	9.75E-04	1.72E-05
5000	2.51E-03	3.58E-05	9.14E-04	1.54E-05
6000	2.28E-03	3.24E-05	7.64E-04	1.40E-05
7000	1.94E-03	3.08E-05	6.90E-04	1.38E-05
8000	1.57E-03	2.90E-05	7.27E-04	1.31E-05
9000	1.50E-03	2.74E-05	6.68E-04	1.27E-05
10000	1.36E-03	2.64E-05	6.94E-04	1.23E-05
50000	1.04E-03	1.30E-05	5.03E-04	8.14E-06
100000	7.14E-04	1.04E-05	4.24E-04	7.44E-06
500000	4.28E-04	7.81E-06	3.46E-04	7.03E-06
1000000	4.33E-04	7.40E-06	3.49E-04	6.99E-06
1500000	4.37E-04	7.22E-06	3.63E-04	6.98E-06
2000000	4.14E-04	7.13E-06	3.57E-04	6.95E-06
2500000	4.14E-04	7.08E-06	3.62E-04	6.94E-06
3000000	3.15E-04	5.99E-06	2.95E-04	5.88E-06
3500000	2.11E-04	3.92E-06	1.91E-04	3.82E-06
4000000	1.22E-04	2.28E-06	1.11E-04	2.21E-06
4500000	6.67E-05	1.04E-06	4.95E-05	9.80E-07

Figure 13 below gives a plot of the average distances above. Since each replication set is compared to a 5,000,000 set that includes all previous replications, the average distance must converge to zero, since at 5,000,000 identical sets would be compared. In fact, there still is some random variation. Based on this plot, extrapolating the trend line out to 5,000,000 replications, we

expect that comparisons to what we will use as a "definitive" data set will still have a random error of about 7E-06. Given the limits on computation time, we decided that this degree of convergence would have to be acceptable. We recognize that the final stochastic reference generated this way will still be only an approximation of the actual outcome set MOE plot. Similar reference sets were also developed for the 4 and 8 division scenarios and for selected other event sets.

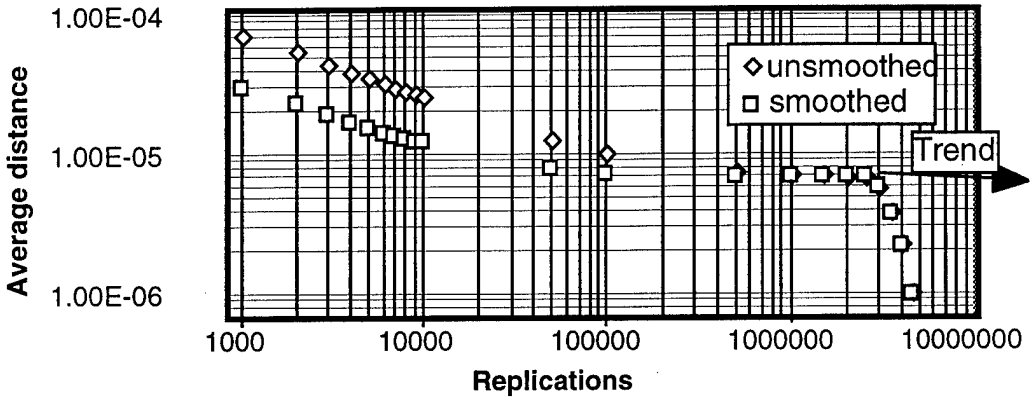


Figure 13 Convergence of MOE Plots with Increasing Numbers of Stochastic Replications

We would expect that the average error from the true outcome space MOE plot would be fairly small. This can be roughly estimated. Only about a fifth of the 10,000 cells of the MOE plot are occupied, and so for the 5M run space each typically is the destination for 2500 trajectories for a total probability of .0005. Each trajectory has a sigma squared of .0004, so that a typical cell will have a sigma of 45 in trajectory count (1.8% of the number of trajectories), or about 9E-6 in probability. The average difference from the true MOE plot will be close to that size, but also includes the 80% of the cells that have no trajectories (and hence close to zero difference), so that overall the average error should be about 2E-6. The average smoothed (3x3) error should be less, about 1E-6. This estimate required many assumptions that are only roughly true, but should give a notion of the expected error. The expected error of 7E-6 estimated from Figure 11 would appear to be conservative.

At this point in the project we appreciated the need for a multitrajectory execution mode that is not so limited by memory as our current techniques, all of which keep time consistent across the simulation (and thus can be called a "breadth first" exploration of the outcome space). We need a "Depth First" Multitrajectory execution mode in order to generate more correct (or representative) outcome space plots against which comparisons can be made. The CPU speed would still be a limitation, but the memory limit would no longer be a practical constraint. The "depth first" technique would require the simulation to have a simultaneous number of states equal to the largest number of events in a trajectory plus one. This is a much smaller limit, and would not be a practical problem until we get to much larger numbers that we have to this time. On the other hand, it is possible to generate impossibly long run times, so even in the depth first approach some means of substituting stochastic (or deterministic) for multitrajectory resolution is needed. The "depth first" technique was not available in time to contribute to the analytic results reported here.

A final remark: The scenarios generated proved to have a "response range" which, at least in the MOE projection, is very well behaved. We see no sharp edges or other evidence of significant small scale nonmonotonicity, despite the discrete nature of the event domain. In some outcome spaces to be described later (for deterministic decisionmaking) we do see large scale nonmonotonicity, but not the kinds of dramatic swings of outcome characteristic on nonlinear deterministic systems operating in a regime of chaos.

Results

We were able to collect extensive data for three different scenario sizes, of 2, 4, and 8 divisions size. For each of these scenarios, runs were made of various sizes, and distance comparisons made to the "definitive" MOE plot generated from 5,000,000 of stochastic replications. Because of memory limitations, the Multitrajectory results are not for single multitrajectory runs, but for a collection of 1000 trajectory runs having different seeds. With Multitrajectory Choice Policy 6, when the limited state budget is exhausted, the simulation runs in stochastic mode. So, this really does not do full justice to the multitrajectory approach. Runs made with larger scenarios of 16 and 32 divisions did not afford opportunities for a valid comparison due to lack of resources to make enormous numbers of stochastic runs. Also, a set of runs was also made in which only C2 events were multitrajectory, and all other events were deterministic. This would be an alternative way to budget trajectories, which assumes the C2 events are the ones which are most important. However, this means that certain trajectories (those associated with the alternative outcomes of other events) will not be in the sample at all. We also made runs in which all events were stochastic or multitrajectory except decisionmaking, since the decisionmaking events seem to have a decidedly different character from the others.

Early data collected for the 2 division scenario are summarized in Table 2 below. The 2 division scenario includes 30 units. All the maneuver units, and most of the other units, become involved in the battle, in contract to the hand crafted 40 unit scenario that was used in earlier studies, in which somewhat less than half of the units enter combat. This data was produced using a 3x3 smoothing prior to making the comparisons. At the time this data was collected, probabilities for the multitrajectory case were not being treated quite correctly. This was found to be particularly a problem when between the hard state limit for the multitrajectory run and the "soft state limit" beyond which low probability runs were resolved stochastically while higher probability trajectories remained multitrajectory. Later work used Choice policy 4 rather than choice policy 6 to avoid this problem.

Table 2 Initial Distance Results from Stochastic, Multitrajectory Runs for 2 Division Scenario

Replications	All events stochastic	All events policy 6	Decision policy 6
1000	2.99E-05	3.79E-05	1.24E-04
2000	2.35E-05	2.60E-05	1.21E-04
3000	1.96E-05	2.20E-05	1.21E-04
4000	1.72E-05	2.21E-05	1.21E-04
5000	1.54E-05	2.02E-05	1.21E-04
6000	1.40E-05	1.91E-05	1.20E-04
7000	1.38E-05	1.93E-05	1.20E-04
8000	1.31E-05	1.83E-05	1.20E-04
9000	1.27E-05	1.80E-05	1.20E-04
10000	1.23E-05	1.87E-05	1.20E-04

For this scenario, for some reason, the Multitrajectory Simulation actually does worse than Stochastic, although the difference is small. The differences in distances are of about the same magnitude as our estimate of error in the definitive outcome set, about 7E-6. The "C2 only" run proved to be a poor substitute for having all events nondeterministic, being a factor of 1.6 worse. The convergence rate to the definitive outcome set with increasing numbers of runs is small. For this scenario, the multitrajectory technique appears to add no value. Making additional replications

in either mode beyond the first thousand or two seems to add little values in convergence toward the reference MOE plot.

It is illustrative to compare the actual outcome plots for this case. Figure 14 shows the stochastic and multitrajectory plots for the 1,000 trajectory case, and Figure 15 shows stochastic and multitrajectory plots for the 10,000 trajectory case. Notice that the multitrajectory outcomes include some with smaller probabilities than are present in the stochastic case, reflecting the fact that some trajectories had less than the average probability at the time when the state limit was reached. This should allow the multitrajectory mechanism to perform better. However, it is also the case that the multitrajectory outcome MOE plot shows more variability in the cells that are toward the middle of the MOE center of mass, which was not expected. It is this variability that seems to account for why the multitrajectory runs grade worse than stochastic. The reason for this remains unexplained at the time this report was completed. There also seems to be a subtle difference in the shape which remains unexplained; the Multitrajectory set shows a bit more of a bulge to the bottom left, which is unexpected if the stochastic and multitrajectory resolutions are (as they should) operating according to the same model. Time has not allowed this seeming discrepancy to be investigated. If there really is a difference, presumably due to some error, the fact that the multitrajectory outcome set MOE plot compares rather poorly to the stochastic reference outcome set MOE plot would be due to this rather than any shortcoming in the Multitrajectory algorithm itself.

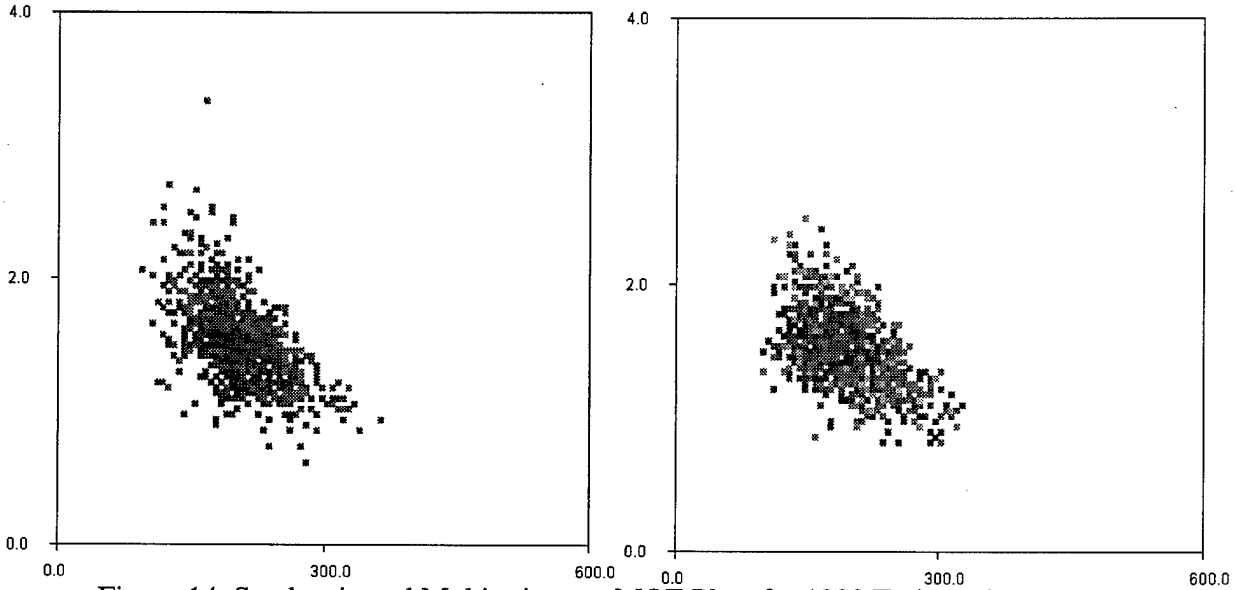


Figure 14 Stochastic and Multitrajectory MOE Plots for 1000 Trajectories

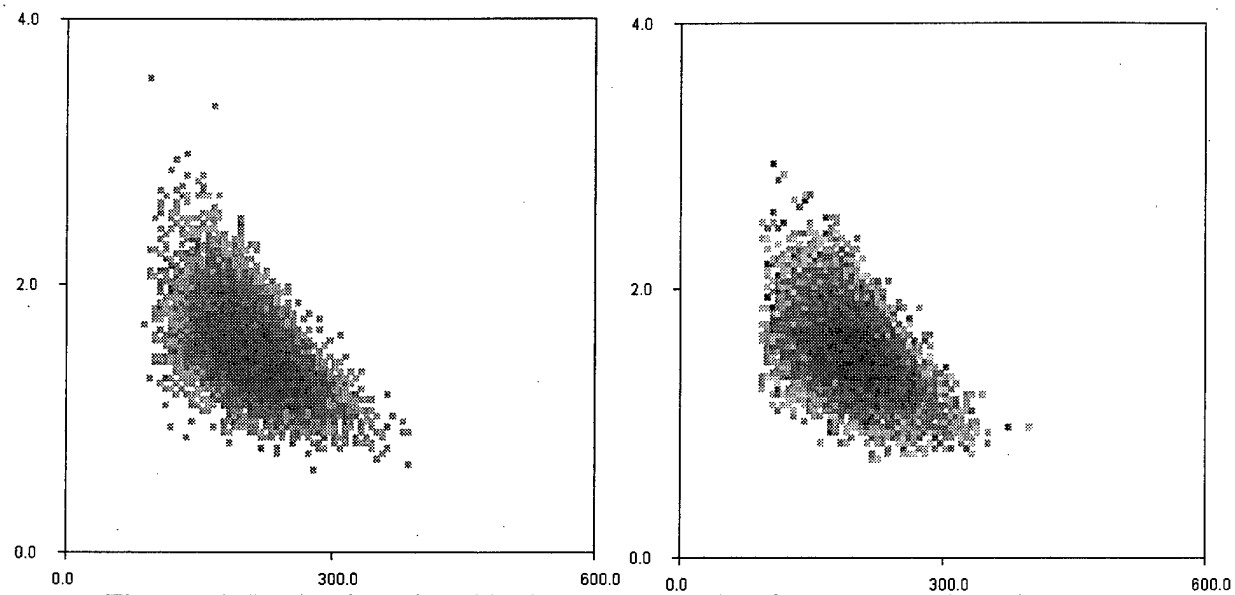


Figure 15 Stochastic and Multitrajectory MOE Plots for 10,000 Trajectories

In order to gain a better understanding of what is going on, we made separate multitrajectory run sets for each of the types of events, hoping to see which, if any, might be dominating the variability result. In Figure 16 we see plots for movement selection events only (with other events being deterministic), in Figure 17 for acquisition only, Figure 18 shows acquisition loss only (not very interesting), and Figure 19 shows decisionmaking only. Note that for most of the cases the figures are very similar, but the granularity effects are particularly pronounced in the decisionmaking case. For individual events, there is no particular difference in shape as had been noted for the plots above.

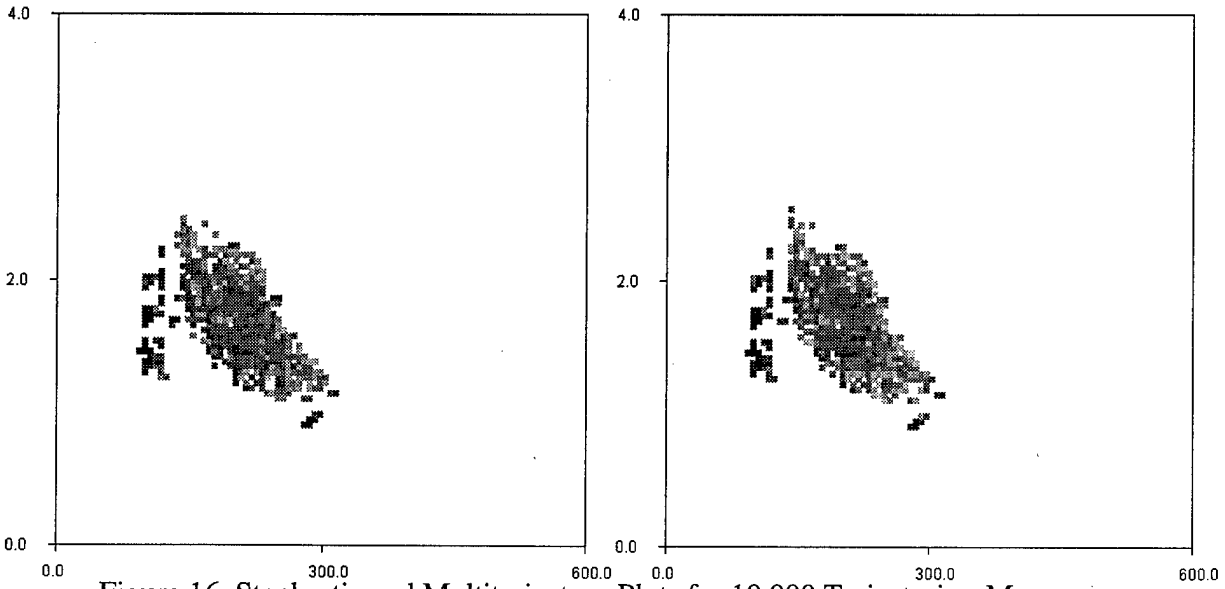


Figure 16 Stochastic and Multitrajectory Plots for 10,000 Trajectories, Movement

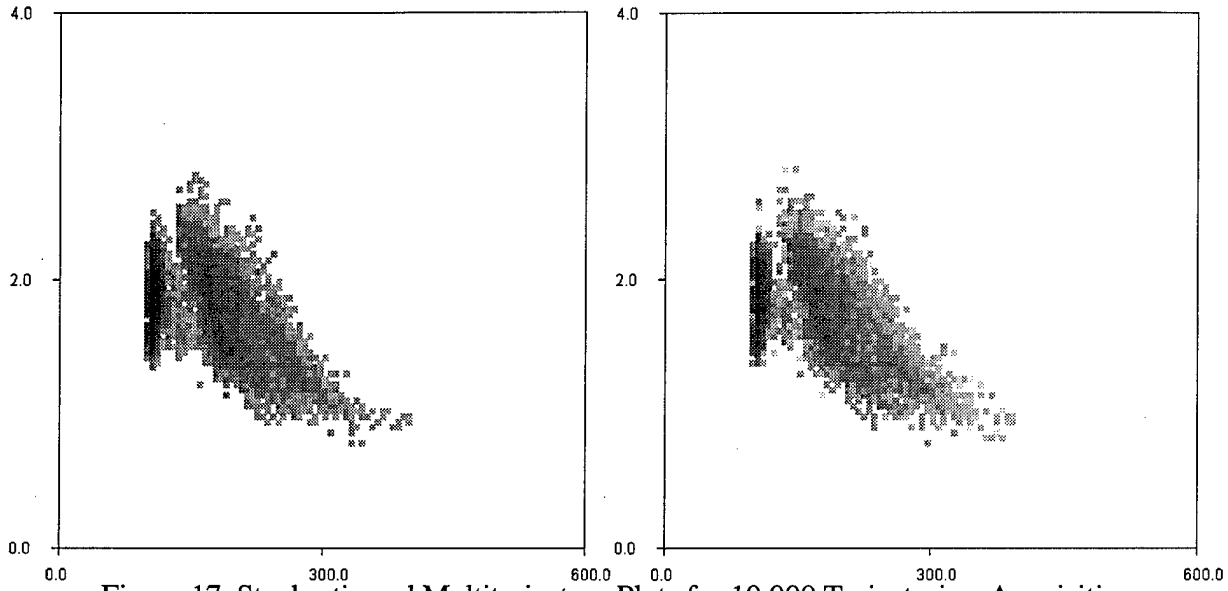


Figure 17 Stochastic and Multitrajectory Plots for 10,000 Trajectories, Acquisition

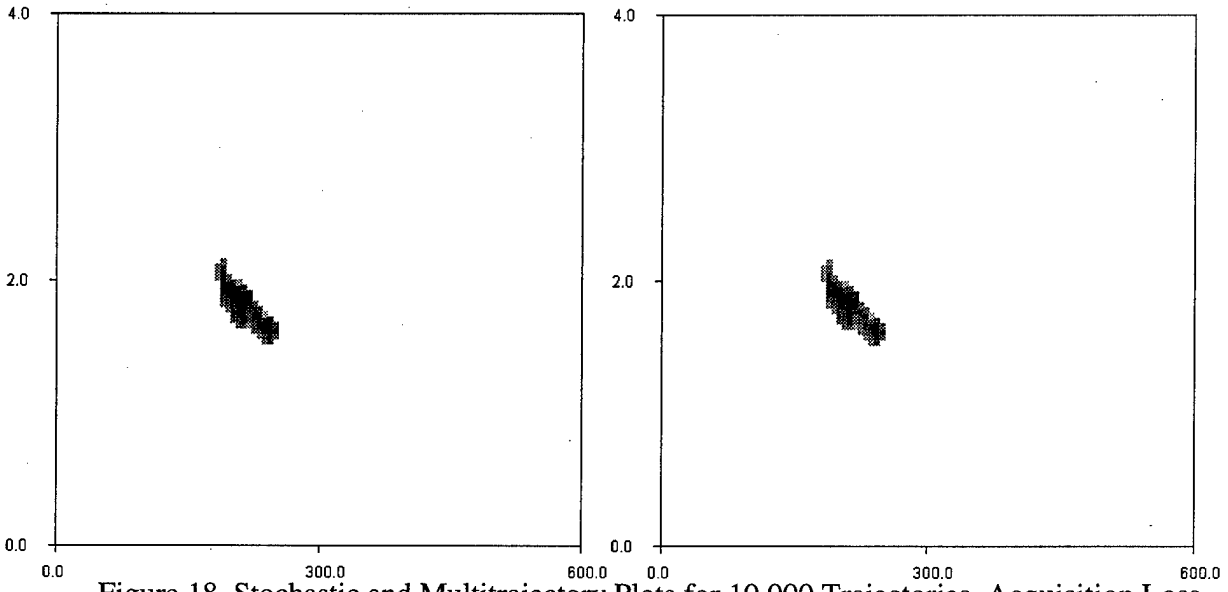


Figure 18 Stochastic and Multitrajectory Plots for 10,000 Trajectories, Acquisition Loss

The decisionmaking event seems to have a notably different effect in the stochastic and multitrajectory modes. This event is different in an important respect. When a decision event occurs for a given unit, in one case the decision is made, and is not tested subsequently because the unit's mode of operation has changed. If the decision is not made, on the other hand, it will usually be revisited later. At each revisit, the probabilities decrease. So a given decision point tends to spawn a chain of side trajectories of ever decreasing probabilities. This gives a much wider variety of probabilities than is usually the case. Movement selection events are singular: once an event occurs, it never recurs. Acquisition events tend to occur few times, because units are usually converging, and soon enter a regime in which probabilities are certainty. Also, once it has detected an enemy unit, a unit fires, making future acquisition against it more likely, giving acquisition events a hysteresis that reduces the event count and prevents probabilities from declining rapidly. The acquisition loss events are relatively unimportant in these scenarios, since units moving apart are usually disengaging and unlikely to affect the battle much anyway. So,

only decisionmaking events are likely to generate very low probability trajectories. In stochastic mode, where the probability is always 1/n, this effect is consequently looks much different.

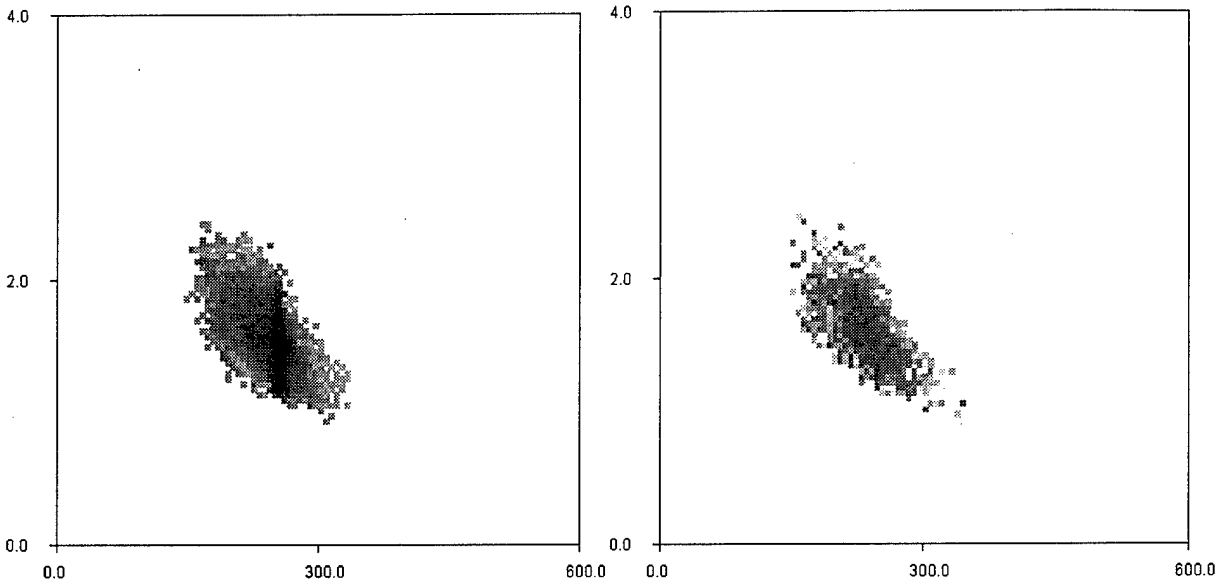


Figure 19 Stochastic and Multitrajectory Plots for 10,000 Trajectories, Decisionmaking

In consideration to this small scale "noise" in the plots from the effects of randomness, a larger smoothing filter of 11 x 11 was tried. This filter would tend to damp out the minor variations. Given the overall smooth nature of the reference outcome set MOE surface, this larger filter would not result in the loss of much information. Table 3 gives the comparison is made of stochastic versus multitrajectory modes with this filter. As can be seen, the stochastic resolution mode still does better.

Table 3 Distance Comparisons, Multitrajectory Runs for 2 Division Scenario, 11 x 11 Filter

	Unsmoothed		Smoothed	11 x 11
Replications	Stochastic	MT,policy6	Stochastic	MT,policy6
1000	7.28E-05	1.01E-04	9.85E-06	1.95E-05
2000	5.52E-05	8.15E-05	8.31E-06	1.44E-05
3000	4.47E-05	6.92E-05	7.74E-06	1.24E-05
4000	4.01E-05	6.13E-05	8.10E-06	1.12E-05
5000	3.58E-05	5.68E-05	7.73E-06	1.09E-05
6000	3.24E-05	5.11E-05	7.33E-06	9.98E-06
7000	3.08E-05	4.86E-05	7.80E-06	1.05E-05
8000	2.90E-05	4.57E-05	7.55E-06	1.06E-05
9000	2.74E-05	4.36E-05	7.25E-06	1.12E-05
10000	2.64E-05	4.25E-05	7.31E-06	1.06E-05

Given the different nature of decisionmaking events, a series of experiments were made in which all events except decisionmaking were made in either stochastic or multitrajectory mode. Decisionmaking events were resolved in deterministic fashion. Figures 20 and 21 show MOE plots for these cases. Note that the response surface is no longer so simple; serious medium scale nonmonotonicity has been introduced. We still do not see small scale nonmonotonicity. Table 4 gives the metrics for distance comparisons. Given smoothing, the multitrajectory algorithm does better, by almost a factor of two. Without smoothing, it does not do as well. Still, this is better

than the results for all events given earlier. It would seem that the poorer performance of the multitrajectory algorithm may have something to do with the decisionmaking events.

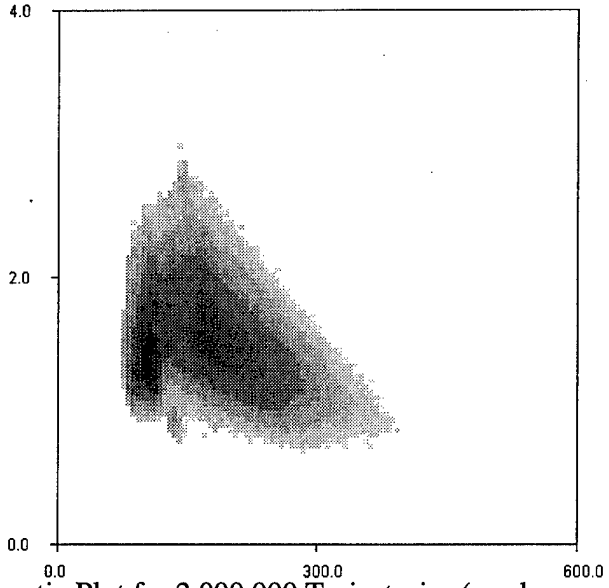


Figure 20 Stochastic Plot for 2,000,000 Trajectories (used as a reference), Two Division Scenario, with No Decisionmaking

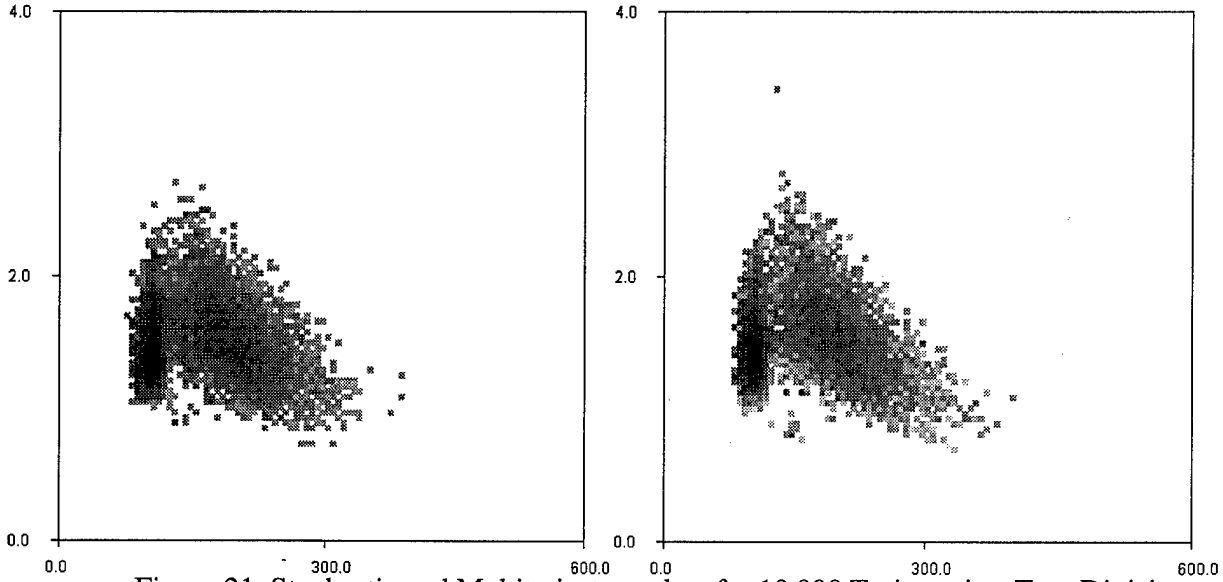


Figure 21 Stochastic and Multitrajectory plots for 10,000 Trajectories, Two Division Scenario, with No Decisionmaking

Table 4 Distance Comparisons, Runs for 2 Division Scenario, with No Decisionmaking

10,000 stochastic replications compared to 2,000,000 stochastic replications:

2.37886E-05 unsmoothed	3.02474E-06 smoothed

10,000 multitrajectory replications compared to 2,000,000 stochastic replications:

3.9005E-05 unsmoothed	1.61944E-06 smoothed
(worse)	(better)

The 4 division scenario is, essentially, two 2 division scenarios taking place side by side. (There are random perturbations of position so that they are not identical battles.) The same series of runs was made for this scenario as for the 2 division scenario. The "definitive" MOE plot that was used as a basis for comparison was a stochastic run of 5,000,000 replications. As for the 2 division scenario, the multitrajectory replications were made using a number of 250 trajectory runs. Figures 22, 23, and 24 show the reference MOE plot and the 1000 and 10,000 replication responses for the stochastic and multitrajectory methods. Table 5 summarizes the distance measurements. All of these runs were made with all (four) event types being resolved in nondeterministic fashion. The largest multitrajectory runs we could make were of 1000 trajectories, so the multitrajectory results are from groups of such runs having different seeds.

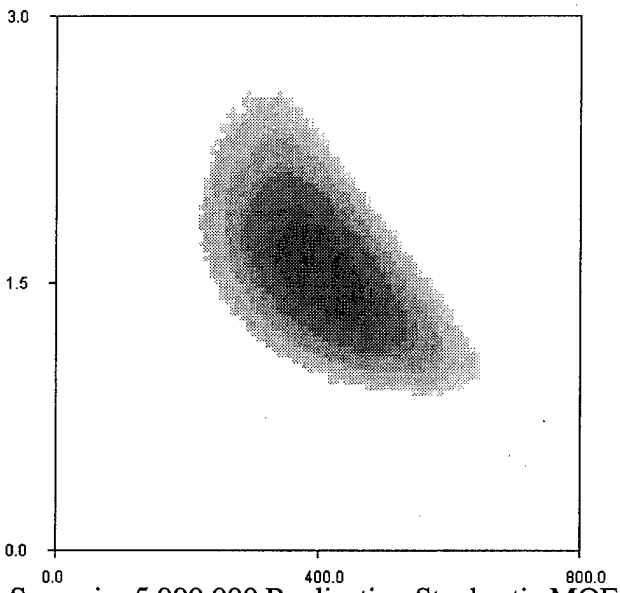


Figure 22 Four Division Scenario, 5,000,000 Replication Stochastic MOE Plot (Reference Plot)

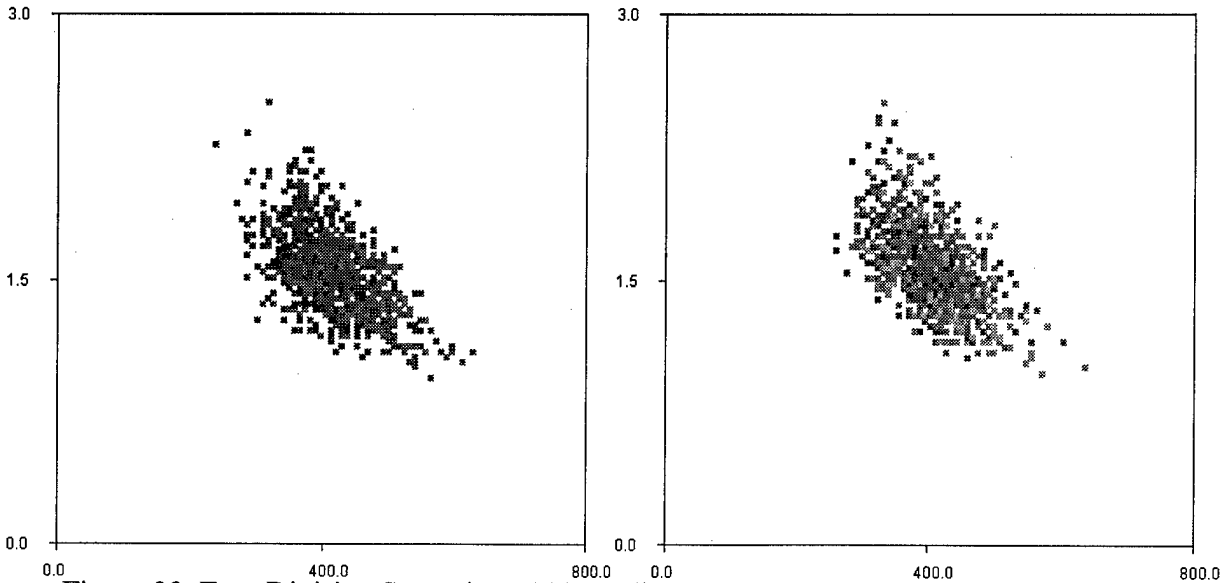


Figure 23 Four Division Scenario, 1,000 Replication Plots for Stochastic, Multitrajectory Resolution.

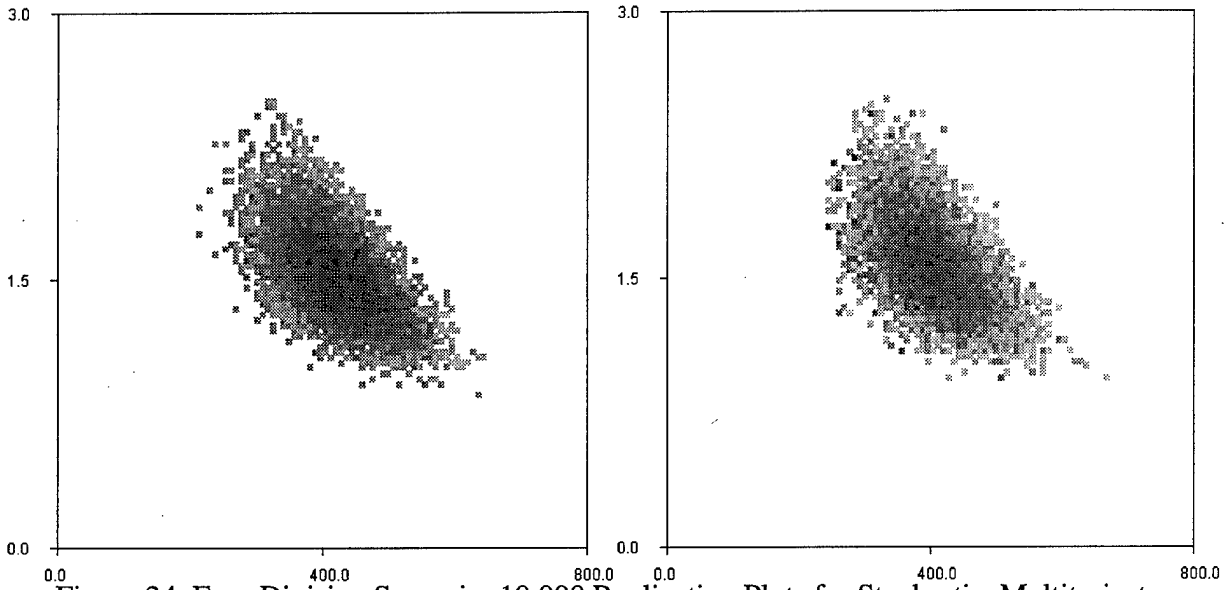


Figure 24 Four Division Scenario, 10,000 Replication Plots for Stochastic, Multitrajectory Resolution.

Table 5 Average Distance Results for Stochastic and Multitrajectory Runs for 4 Division Scenario (With all Distances Calculated Using a 11 x 11 Filter)

	Unsmoothed		Smoothed 11 x 11	
Replications	Stochastic	MT policy4	Stochastic	MT policy4
1000	7.10E-05	0.0001073	1.77E-05	1.78E-05
2000	5.40E-05	8.65E-05	1.88E-05	1.14E-05
3000	4.53E-05	7.39E-05	1.83E-05	1.06E-05
4000	4.15E-05	6.73E-05	1.85E-05	8.00E-06
5000	3.86E-05	6.22E-05	1.90E-05	7.93E-06
6000	3.63E-05	5.84E-05	1.87E-05	6.40E-06
7000	3.44E-05	5.51E-05	1.92E-05	5.58E-06
8000	3.31E-05	5.16E-05	1.95E-05	5.94E-06
9000	3.21E-05	4.93E-05	1.98E-05	5.55E-06
10000	3.10E-05	4.63E-05	1.95E-05	5.02E-06
50000	1.33E-05		7.57E-06	
100000	8.45E-06		3.15E-06	
500000	3.37E-06		9.60E-07	
1000000	2.29E-06		6.72E-07	

The distance metrics are significantly better than for the smaller 2 division scenario. This reflects the fact that the total number of events is about twice as large. In addition, there are twice the number of units over which metrics are calculated. So, there are two compounding "large number" effects that increase the convergence. We also see that now, remarkably, the Multitrajectory version outperforms the stochastic version in the closeness to the definitive outcome set when comparing smoothed data. (The granularity of the multitrajectory MOE plot does worse prior to smoothing.) The difference in performance increases as the number of replications increases. At 1000 replications (or trajectories) the difference is slight. At 10,000 replications, the difference is more than a factor of 2 in favor of the multitrajectory version. The "C2 only" replications are an even poorer approximation in this scenario than before, with the difference ranging from a factor of 2.6 up to almost 8.

As with the two division case, we suspected that the small scale variability was causing excess noise at these small numbers of replications. A separate set of comparisons were made using an 11 x 11 filter. For these comparisons, we used choice policy 4, which provided a simple hard limit on the number of trajectories. These results are tabulated below in Table 7.

The 8 division scenario is simply four 2 division scenarios side by side. Again, there are random perturbations so that each of the division battles, while having the same basic structure, is different in details. Table 5 gives the data generated from these runs. As with the 2 and 4 division scenarios, a 5,000,000 replication set provided the reference MOE plot, and the multitrajectory replications were a set of separate 250 trajectory runs. (There was not time for the C2 only runs.) Figure 25 shows the reference MOE plot, and Figure 26 shows comparisons for 10,000 replications.

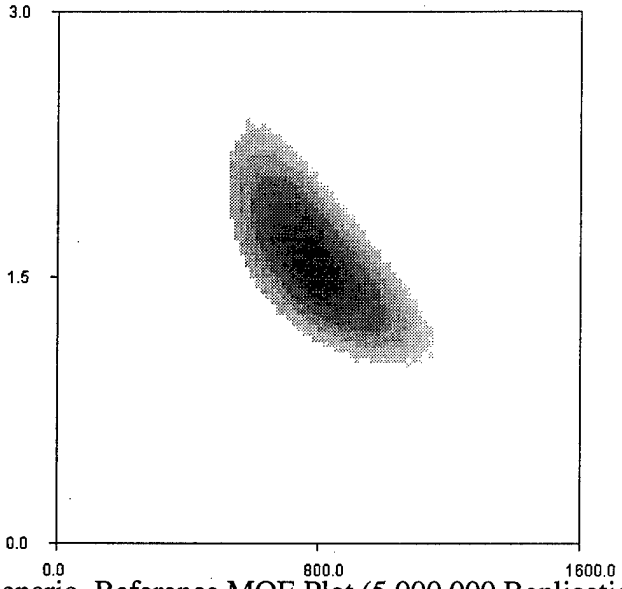


Figure 25 8 Division Scenario, Reference MOE Plot (5,000,000 Replications)

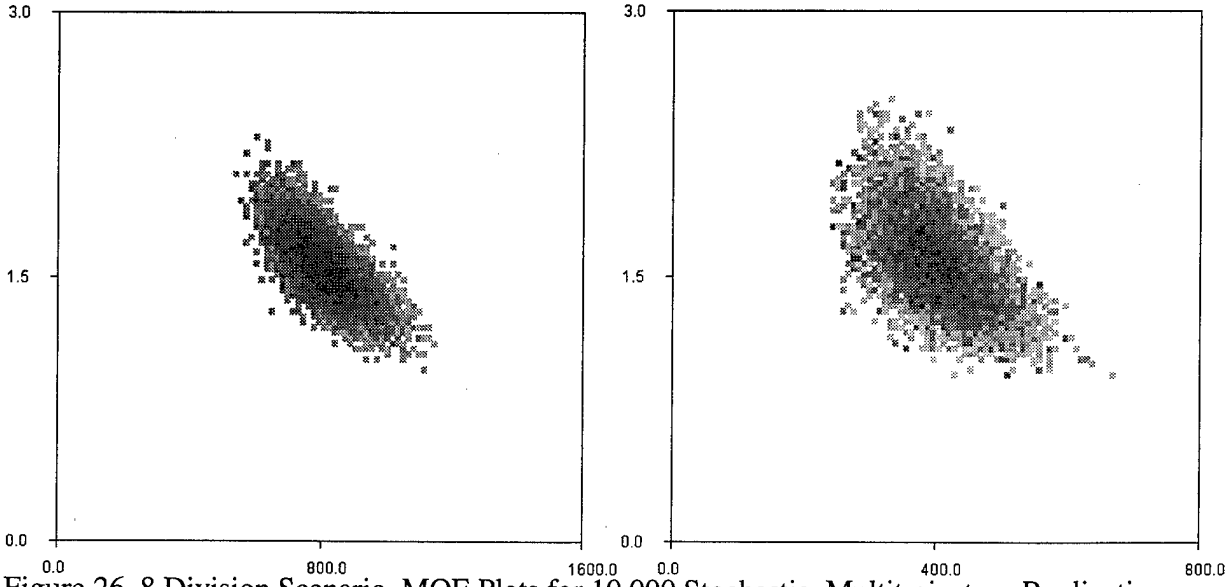


Figure 26 8 Division Scenario, MOE Plots for 10,000 Stochastic, Multitrajectory Replications

Table 5 Distance Results from Stochastic, Multitrajectory Runs for 8 Division Scenario

Replications	Unsmoothed		Smoothed, 11x11 filter	
	Stochastic	MTpollicy4	Stochastic	MTpollicy4
1000	5.85E-05	2.58E-05	2.58E-05	2.71E-05
2000	4.61E-05	2.61E-05	2.61E-05	2.04E-05
3000	4.02E-05	2.62E-05	2.62E-05	1.87E-05
4000	3.75E-05	2.62E-05	2.62E-05	1.59E-05
5000	3.58E-05	2.61E-05	2.61E-05	1.23E-05
6000	3.48E-05	2.62E-05	2.62E-05	1.23E-05
7000	3.41E-05	2.63E-05	2.63E-05	1.23E-05
8000	3.37E-05	2.65E-05	2.65E-05	1.23E-05
9000	3.32E-05	2.64E-05	2.64E-05	1.23E-05
10000	3.24E-05	2.60E-05	2.60E-05	1.23E-05
50,000	2.81E-05		2.54E-05	
100000	2.77E-05		2.54E-05	
500000	2.77E-05		2.57E-05	
1000000	2.75E-05		2.56E-05	

In this set of runs, the Multitrajectory mechanism does remarkably better than the purely stochastic approach, by a distance factor of about 2 as the number of replications increases. Even in the unsmoothed comparisons the multitrajectory mechanism does better. Thus, 1000 multitrajectory replications gets us closer to the definitive outcome set than 10 times the number of stochastic replications. Yet, increasing the number of multitrajectory replications yields little additional advantage. (Larger multitrajectory runs, of more than 250 trajectories, should help, though.)

Although we were not able to make distance comparisons to as large a reference MOE, we have MOE plots for the 16 Division Case, shown in Figures 27 and 28 below. As can be seen, the small 1,000 replication Multitrajectory case exhibits quite a bit of granularity; much more so than the comparable sized stochastic replication set. The 30,000 stochastic run set was too small to serve as a definitive reference for meaningful analysis. Plots for the 32 division case were not available in time for this report.

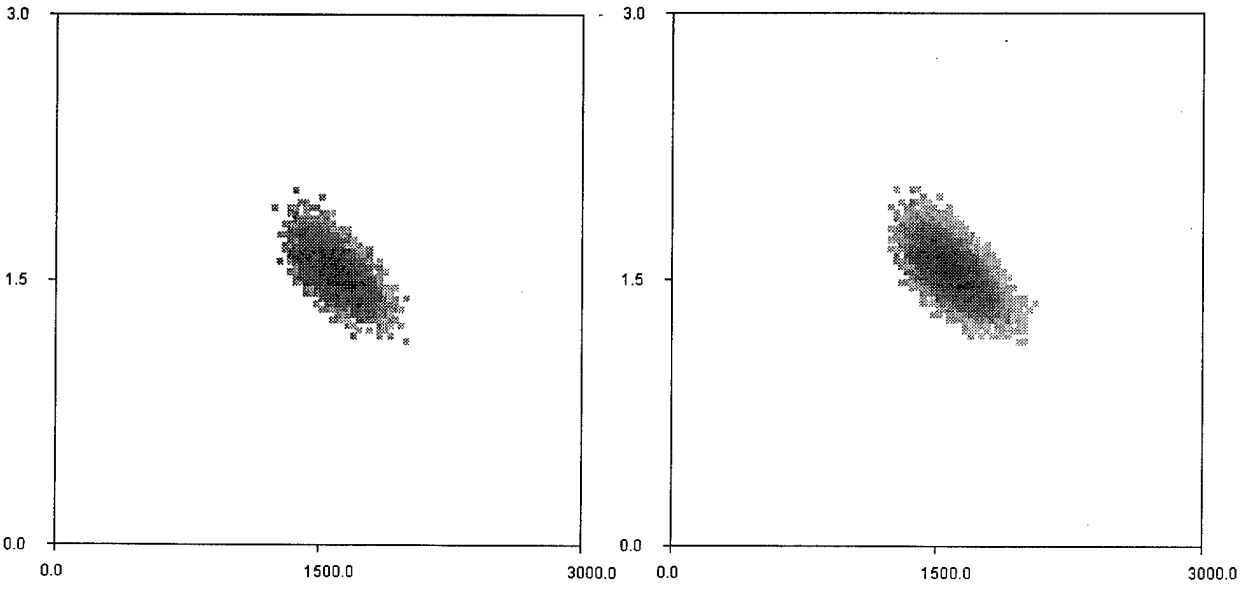


Figure 27 16 Division MOE Plots, Stochastic 10,000 and 30,000 Replications

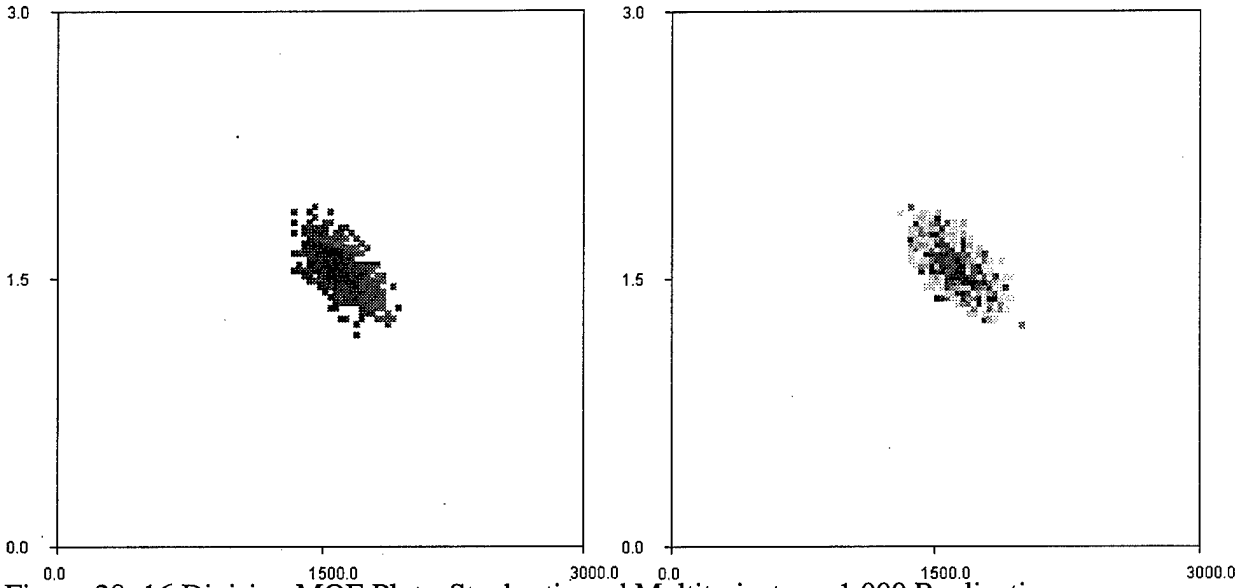


Figure 28 16 Division MOE Plots, Stochastic and Multitrajectory, 1,000 Replications

Conclusions

The results obtained are unsatisfying, having a number of features we are unable to explain at this time. We are continuing to check the methodology and analysis software, especially for possible flaws having to do with the generation of random numbers. So, at this time we are reluctant to affirm any strong conclusions.

If the results are taken at face value, then they seem to indicate that increasing scenario sizes make the Multitrajectory approach more advantageous. For the smallest (2 division) scenario analyzed, we saw no advantage, or even a slight disadvantage. For the 4 division scenario, the Multitrajectory approach had a significant advantage. By interpolation, we would expect that a 1200 replication stochastic run would match the 1000 replication Multitrajectory run set in terms of achieving the same distance to the definitive outcome set. For the 8 division scenario, even 10,000 stochastic replications gives a greater distance than the 1000 replication Multitrajectory run, and the rate of convergence is so slow that the number of stochastic replications needed for equivalence is probably huge.

There are reasons to distrust this data. We do not understand why the Multitrajectory results for the 2 division scenario should be worse than stochastic. The Multitrajectory performance should be bounded above by stochastic performance. There may be some sort of error that results in the multitrajectory and stochastic outcome sets being different, as seen in the slightly different shapes of the outcome MOE plots. Possible residual errors in the definitive outcome set are too small to account for this. In the small 4 unit scenario (scenario_1b), where multitrajectory runs can be exhaustive or nearly so, the number of stochastic runs necessary to even approach the multitrajectory result is quite large, certainly more than a factor of 10 (judged subjectively, we admit).

Recommended Future Modifications to "eaglet" or analyses:

1. Alternative decision model with "hysteresis" (A given situation causes only one testing of a rule, rather than successive testings and trajectory generations at successive time steps.)

Status: Implemented, but not exercised in runs to date.

2. Explicit Decisionmaking by HQ units
Status: An easier alternative was that individual maneuver units have contingencies reflecting upper level C2 options. We did that instead to save effort. The more general technique provides a more reasonable approximation to how analytic grade simulations may operate.

3. Upper level "planning" (which not planned for this project)
Status: The hierarchical scenario generator actually is a partial implementation of this capability, which is needed in the simulation proper to represent reserve organizations above the resolution level. Absence of this limits the scenario length that can be reasonably represented, and restricts the size of upper level organizations unrealistically.

Remaining issues

Some things we really have not addressed in either this or the previous project include:

1. Theory: What can we calculate about multitrajectory behavior and benefits? To date, all work has been more experimental, and focused on the "Can it even be done" question.

2. Characteristics of an Outcome Space: What is important? So far we have merely been making, in effect, efficiency comparisons against stochastic simulation. Can the multitrajectory technique be used to direct the simulation toward "interesting" or "preferred" outcomes?

3. Shared states and lazy state cloning: Currently, new states are generated in their entirety when an event occurs. Yet, at the time of an event, typically only one state variable changes. It may eventually make a difference; it may not. An alternative in which new objects are cloned only when necessary (when differences propagate) would not just be a possible source of greater efficiency. It would also allow much easier state difference evaluation. This improvement will be "hard" but it may also be very worthwhile.

4. Multitrajectory Simulation in the larger study context: Can the multitrajectory technique allow new analysis techniques not available now? These might include marrying traditional simulation through the use of event management to a planner or other goal directed engine, which can participate in the execution control of the simulation. This is a subject for the next effort, to some extent

5. Multiprocessor execution It should be easier than single trajectory parallel simulation.

List of Project Participants

John B. Gilmer, Jr., Associate Professor, Electrical and Computer Engineering, PI

Frederick J. Sullivan, Associate Professor, Mathematics and Computer Science

Eric Werner and Karen Mazuka, undergraduate students.
Both graduated in May, 1998, and subsequently continued to work after graduation until August, 1998. Eric Werner is now a graduate student at the University of Pittsburgh, and Karen Mazuka is employed by Booze Allen in Arlington, VA.

It should also be noted that during the interval between the previous project and this one, Sadeq Al-Hassan received his MSEE in January 1997 based on unfunded work that built upon the results of the previous project. He is now employed by Intel.

Report of Inventions

No inventions are claimed for this project.

Bibliography

1. J.A. Dewar, J.J. Gillogly, and M.L. Juncosa of RAND, "Non-Monotonicity, Chaos, and Combat Models", presented at the 59th MORSS, 12 June 1991, in the Best Working Group session, and winner of Rist Prize. Dewar and his co-authors demonstrate chaotic behavior for a one sector Lanchester square law model in which C2 makes decisions to insert reinforcements from outside the model. Since it runs indefinitely with both sides adding reinforcements, this scenario is able to exhibit strange attractors and other characteristics of a chaotic system.

2. J.B. Gilmer, "Risk Maps: A Look at Chaotic Behavior", presented at the 59th MORSS on 13 June 1991 in WG27 "Measures of Effectiveness". The paper is available from the author. This paper addresses the same phenomenon as the Dewar et. al. paper, but for a more complex system and on a non continuous basis.

3. J.B. Gilmer, "A Search for the Sources of Chaos", presented at the 61st MORSS on 22 June 1993 to WG32 "Advanced Analysis, Technologies, and Applications", on 23 June 1993 to the Joint session of WG15 "Command, Control, and Communications" and WG19 "Measures of Effectiveness", and on June 24 to WG30 "Decision Analysis". The paper follows up the previous reference and is available from the author.

4. John W. Ogren, "Command and Control in the Eagle Simulation", Eagle Concept Papers, Vol. VI, TRADOC Analysis Command, Ft. Leavenworth, KS. This is a representative document describing mechanisms that are of particular interest in the simulations being studied. The Principal Investigator also has in his possession numerous design documents concerning the Corban simulation, including its Command Control representation.

5. J. B. Gilmer and F. J. Sullivan, " Combat Simulation Trajectory Management," *Proceedings of the 1996 Military, Government, and Aerospace Simulation Conference*, SCS, April 1996. This paper describes work done in the preceding project.

Appendix A

The Incoherent Planner:

The basic concept is that the generation of data randomness will be used to ensure a lack of coherence in unit placement, routes, support assignments, and tasks. Several basic assertions below, without any seeming loss of generality, help organize the scenario:

Units:

1. The number of units on each side is equal, with any odd unit being Blue.
2. Blue attacks Red, moving left to right (increasing x coordinate).
3. Both forces deploy perpendicular to the attack axis. (The attack is frontal).
4. The y dimension of the "box" in which each force is deployed is larger than the x dimension. Both Red and Blue boxes have the same size. A 10 unit (total) scenario has boxes 10 km wide by 5 km deep.
5. The y dimension scales linearly with the number of units. (Other options are also possible.)
6. The initial separation of the two sides (the distance between the "boxes") is constant. (A 5 km distance has been used.)
7. Both sides have maneuver (nominally tank), artillery, and HQ units.
8. The proportions of different kinds of units will be constant. (We use 50% maneuver, 30% artillery, 20% HQ unit types)
9. Maneuver units should be placed closer to the enemy; artillery and HQ units will generally be farther back. A triangular probability density function is used. For maneuver units, there would be a zero probability of being at the back boundary of the box, and twice the mean probability at the front of the box. For artillery and HQ units, the triangular distribution would be in the opposite direction.
10. Unit numbering is arbitrary from the view of simulation mechanics. For command and support purposes, the ordering can be useful. For each side, the first unit is an HQ unit. After that, units are drawn randomly. Maneuver and artillery units are assigned to the immediately previous HQ. If an HQ follows another HQ, it is subordinate to that HQ, unless this would reach below brigade level. In that case, it is subordinate one level further up. Also, if such an assignment gives the HQ more than 6 subordinates, it is bumped up a level in the hierarchy.
11. All maneuver and artillery units are to be "battalion" echelon (echelon 4). The HQ units should start at echelon 5 for less than 10 units, 6 for 11 to 100 units, etc.
12. Blue maneuver units start with strength 100, Red with strength 70. Artillery all start with 20, and HQ units with strength 5.
13. All Blue units should start with an axis of 90, and Red units with an axis of 270.

Tasks:

1. All HQ units have an intent, task type, and operational activity of "HQoperate" (9).
2. All artillery have an intent, task type, and operational activity of "Arty-supt" (8).
3. Artillery units support the previous maneuver unit of their side in the scenario. (This would be the most recently created maneuver unit, if any.)
4. All Red maneuver units have an initial task type, intent, and operational activity of "defend" (1), with the objective being their current location.
5. All Blue maneuver units which start in the rear half of the Blue "box" have an initial task type, intent, and operational activity of "defend" (1), with the objective being their current location.
6. On a random draw, half of the Blue maneuver units starting in the front half of the Blue "box" start with initial defend tasks as above.
7. The other Blue maneuver units in the front half of the box receive an attack task. The objective is to be within the Red box. The x dimension of the objective is to be

chosen with a uniform distribution, and the y dimension from a triangular distribution centered on the y location of the unit and having a width equal to the distance from the unit to the objective in the x dimension. (Routes are needed.)

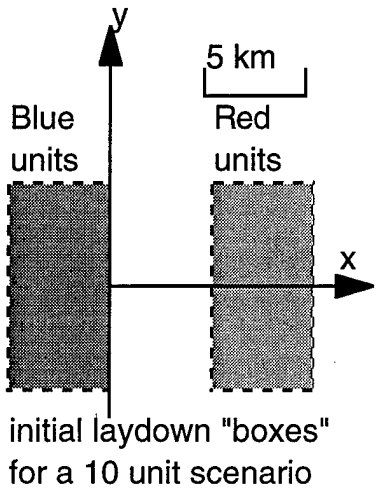
Routes:

Only maneuver units having attack orders will have routes.

1. A route pattern will randomly be selected from a list, with the start point and objective given by the unit's initial location and task objective.
2. Other points along the route will be subject to random variation in x and y dimensions, about the nominal placement point, of + or - 25% of the distance between the start and end points.
3. Route numbers are assigned corresponding to the unit number. Link numbers are sequentially allocated.
4. Trafficability values are randomly selected between .5 and 1.0.
5. Route branch probabilities (where applicable) are determined by randomly choosing one branch as having a probability of .2 to .8. Use route topologies having only two way branches.

Illustrations:

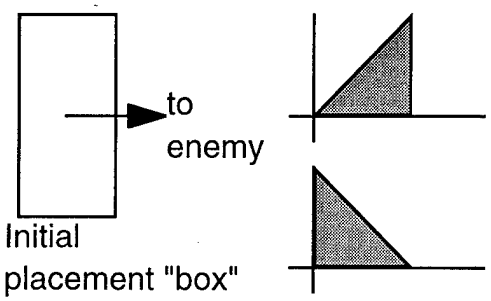
Figures 1 through 4 below illustrate the operation of the "incoherent" planner:



The size of the two initial placement boxes are scaled linearly with number of units. (Other scaling options have not been used.)

Units are placed in the boxes uniformly in y and according to the triangular distributions below in x.

Figure 1 "Boxes" in which Blue and Red units are placed by the incoherent planner



Unit mix is according to a set of parameters, e.g. 50% maneuver.

Figure 2 Blue and Red unit disposition distributions used by the "incoherent" planner

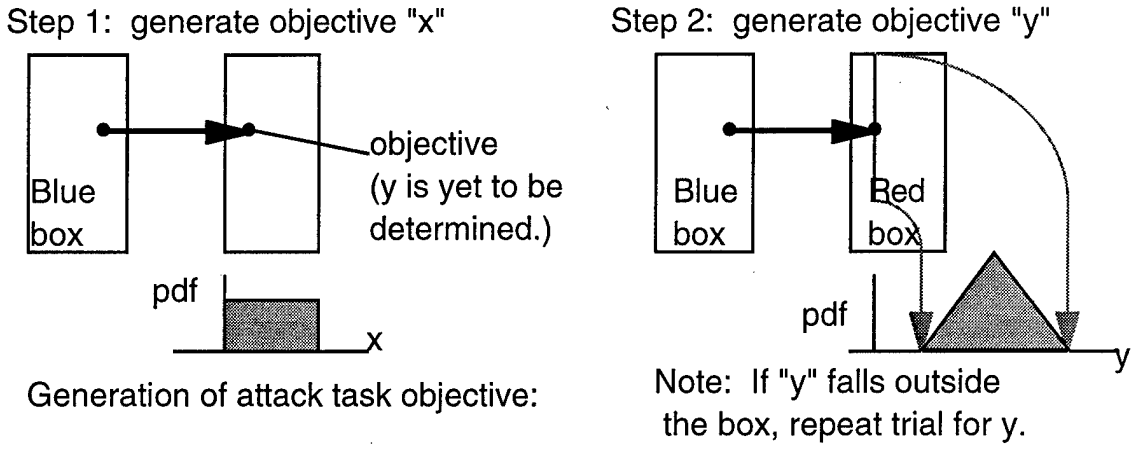


Figure 3 Selection of objectives for Blue maneuver units by the incoherent planner

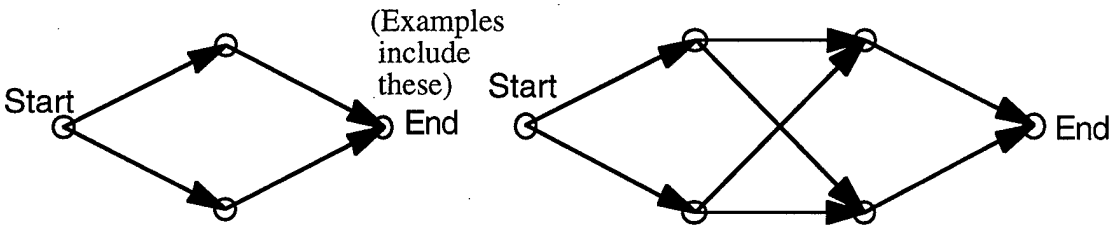


Figure 2 Routes drawn from set of templates, used by the incoherent planner

As units are generated from 1 to n, Blue units are generated first, then red units. The unit type is randomly determined. Maneuver and artillery units are assigned to the most recently generated HQ unit, maneuver units get support from the most recently generated artillery unit.

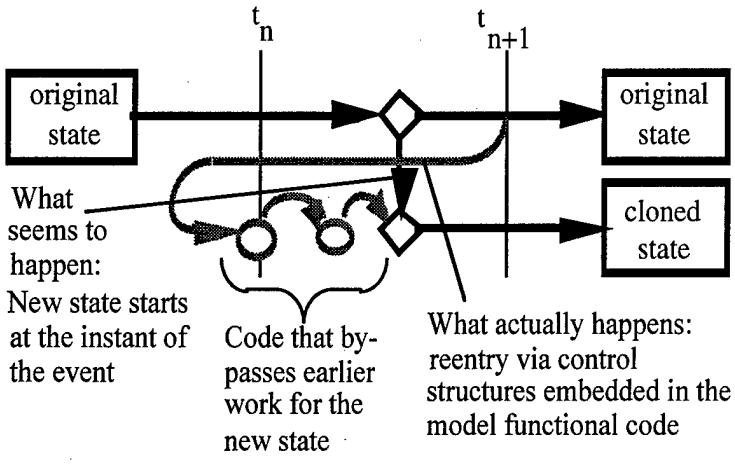
The incoherent planner was built and tested, but was completed after the hierarchical planner. Time did not permit the experiments for which it was intended.

Appendix B

Alternative Multitrajectory Techniques

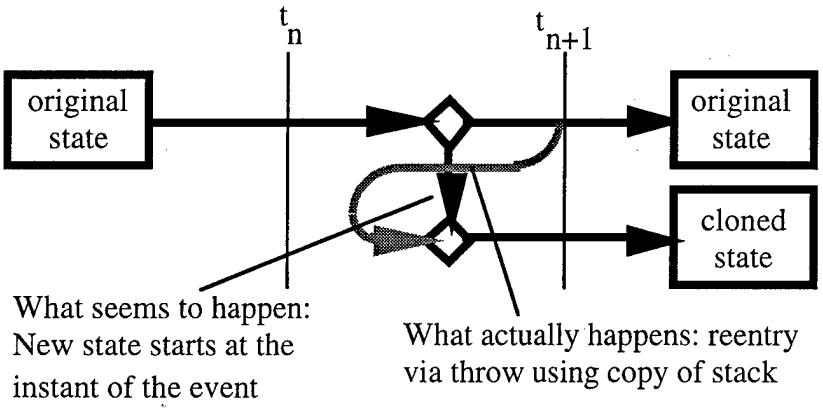
In the course of work on this and a related project (Course of Action Analysis, for SAIC) some alternative methods of implementing Multitrajectory Simulation have been developed. These are illustrated in this appendix. A fuller treatment of this topic is available as a separate paper. [reference]

1. Gilmer's Method: In the original prototype, multitrajectory execution was embedded in the model code. At the top of every routine, it was necessary to provide for reentry for the case of a state that was resuming execution at the point where it was created in an event resolution. The code to do this was messy, and became much more so as new variations in event resolution techniques were developed.



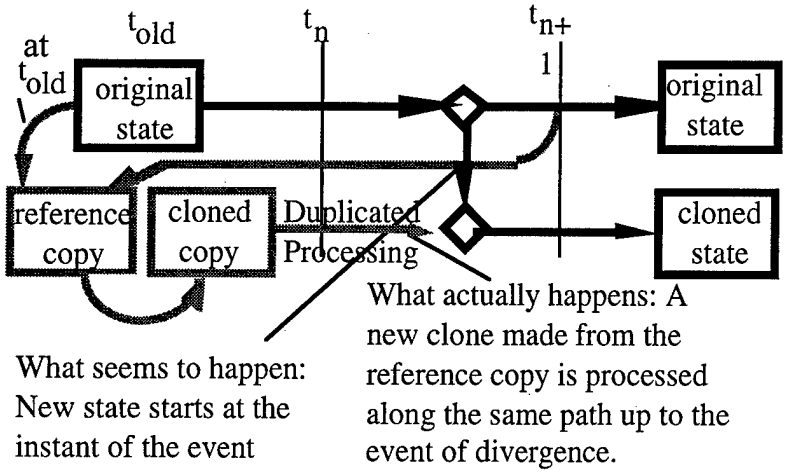
Gilmer's (original) method

2. Sullivan's Method: Reentry is accomplished by copying the stack when an event occurs, and performing a throw (a C longjmp) when the newly created state begins execution. To the model functional code programmer, it appears that a chooser function is called once but returns multiple times: once for each choice. This technique hides the messy details in base classes, and is much cleaner than the earlier method. However, there is a "this" problem: "this" as well as any other pointers restored from the stack point to objects in the wrong state. This problem can be managed. This technique requires a small amount of machine dependent code.



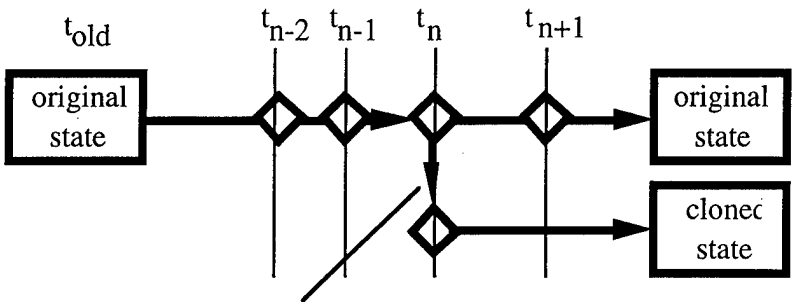
Sullivan's (magic) method

3. Koch's method: The simulation trajectory is copied periodically. When an event occurs, the new trajectory starts not at the instant of the event, but with the most recent copy. The trajectory is repeated up until the event takes place, then the alternate resolution is made. This technique is more straightforward, but requires more memory and computation than either of the others. This technique is being implemented as a senior project by Wilkes University students.



Koch's Method

Burlingtom Method: This technique was used by SAIC / Burlington personnel to implement a multitrajectory version of the prototype simulation, "ACP". The simulation is organized in discrete event fashion, and only events occurring in this sense can be multitrajectory. This restriction means that events never occur embedded inside the hierarchy of the model functional code. This does restrict the style of a simulation, and force the use of scheduled discrete events where that would otherwise not be necessary, at some cost in event processing overhead.



What actually happens and what seems to happen are the same, since the events of the simulation at the dispatch (DES) level are the multitrajectory events (and any that may be multitrajectory). Events embedded deeply in model code are prohibited.

Burlington Method

In addition, a second method developed by Sullivan based on reorganizing the simulation code to a tail recursive form was developed, but has been tested only with a trivial scale "random walk" simulation.

Appendix C

Published and Working papers

The following working papers were developed during the course of this project, or in work closely related to this project. Copies of these papers can be found on the project web site: hobbes.wilkes.edu/~mts. This listing does not include the working papers from the previous project, which are also available at the same web site.

"Method for Generating Large "eaglet" Scenarios," October 27, 1997

"eaglet" requires input of data for each and every unit to be represented in the scenario. This was manageable for the scenarios used in the earlier stages of the project, which had 4 or 40 units. Even for 40 units, ensuring correctness was a problem. For larger scenarios, preparation of individual unit input is practically not possible. Some method for rapidly generating scenario data, is necessary. This working paper outlines the approach proposed.

Decisionmaking for Upper Echelon Units for "eaglet", October 28, 1997

In the current version of "eaglet" most decisionmaking has been on the unit level, with individual units making decisions based on their own state. In some cases, rules hard wired into the code were used for decisions that reflected the activities of the upper echelon decisionmakers. A less ad-hoc method will be needed for a scalable version of "eaglet", since units and organizations will be generated automatically. This paper outlines the problem more specifically, and discusses a proposed solution.

"Plans for Improvements to Multitrajectory Management ," October 31, 1997:

Discusses objectives for the current project is to improve the management of trajectories, especially as the simulation scenario size grows. The effort on improved management includes several components. These include the metrics used to measure the "distance" between states, especially as the metric may be important as a predictor of future convergence or divergence with respect to an MOE (Measure of Effectiveness) of interest. A second component is the identification of choice policies that will preferentially select interesting or important trajectories for continuation, even in the face of numerous choices. This working paper outlined some of the concepts to be developed over the course of this project, and some of the issues to be resolved.

"Incoherent Planning for "eaglet" Scenario Scaling," January 26, 1998

A more expedient, if somewhat less realistic, scenario generation approach was needed. Random assignment of units and task objectives is developed such that a scenario of arbitrary size can be generated.

"Eaglet example execution, small scenario," June 3, 1998 (file eaglet scenario_1b script)

This document illustrates the execution of the multitrajectory simulation "eaglet" using a small scenario of just four units.

"Eaglet example execution, larger scenario," June 3, 1998 (file eaglet scenario_2b script)

This document illustrates the execution of the multitrajectory simulation "eaglet" using a 40 unit scenario that elaborates on that of the "small scenario". In the small scenario two Blue units attack a Red unit, and another Red unit makes a counterattack. This document assumes that the reader is already familiar with that smaller scenario.

"Compromise Modifications to 'eaglet'," June 26, 1998

Several compromises were necessary in the improvements planned for "eaglet." Given the difficulties with getting the full "Sullivan's Method" version of "eaglet" running properly, modifications that required substantial changes, especially changes that would require

modifications to control structures, needed to be avoided. Alternatives are described in this working paper.

"Template Based Scenarios for 'eaglet'," July 25, 1998
This working paper describes the generation of large scenarios for the "eaglet" simulation using the "Hierarchical Planner." It illustrates the template based scenarios by example.

In addition to the working papers above, the following paper was submitted for publication and presentation at the 1998 Winter Simulation Conference in Washington, D.C. Appendix B contains a greatly shortened version of the results reported in this paper.

John B. Gilmer Jr. and Frederick R. Sullivan, "Alternative Implementations of Multitrajectory Simulation," December, 1998.

Abstract: Multitrajectory Simulation allows random events in a simulation to generate multiple trajectories, and the set of trajectories is explicitly managed. The original prototype combat simulation used to demonstrate and test the concept used code embedded in the functional modules, e.g. those that implement "move", "shoot", etc. A much improved method provided a class library that hid the messy details. When a random choice is made, a random choice method appears to return twice (or more) for a given call, once in the context of the original state, and once each for the new trajectories. These techniques both have significant shortcomings. For example, the second (more preferable) technique really needs to be able to overwrite the C++ "this" variable, an option unavailable on C++ compilers. Also, machine specific code is needed to effect the reentry into the chooser. In Java, this requires the unattractive option of modifying the abstract machine. Since these issues surfaced, three additional implementation techniques have been developed. These include periodic copying of states to provide reference copies and duplication of trajectories up to the branch point, Reformulation of the simulation to a discrete event style, and reformulation of the simulation into a tail recursive style. This paper reviews these various techniques, and illustrates that the multitrajectory concept is not dependent on the particular limitations of any one method. Each has its advantages and disadvantages. All except one of these techniques have been prototyped on at least small scale.

Appendix D

Concurrent work for SAIC (COAA Project)

During the period of performance of this project, Fred Sullivan and John Gilmer also provided consultation to SAIC, Inc. This work was performed on the DARPA sponsored "Course of Action Analysis" project, through the SAIC Arlington office. Wilkes's involvement was limited to the examination of multitrajectory simulation as a possible method for doing "Range of Outcomes" generation. The work performed included:

1. Consultation on project as a whole

2. Modification of "eaglet" to generate data for displays

3. Runs to illustrate "Range of Outcomes" possibilities

4. Examination of alternative implementation techniques

For all of this work, the Old "Mac" (Gilmer's Method) version of "eaglet" was used. So the SAIC project did not benefit much from concurrent work performed on this project funded through Army Research Office. This project benefited from some tools (most importantly, a graphics "eaglet") that aided in development and debugging.